Selenium Tutorials 32 Best Free Selenium Training Tutorials

Posted In | Automation Testing, Selenium Tutorials | Last Updated: "December 14, 2016"

After hundreds of requests from STH readers, today we are finally launching our FREE Selenium Tutorial series. In this Selenium training series we will cover all Selenium testing concepts and its packages in detail with easy to understand practical examples.

These Selenium tutorials are helpful for beginner to advanced level Selenium users. Starting from the very basic Selenium concepts tutorial, we will gradually move on to the advanced topics like Framework creation, Selenium Grid and Cucumber BDD.

Note: We will be increasing our article posting frequency for this series. Please don't miss any tutorial. Keep track of all the tutorials by bookmarking this page as we will keep updating it with links to all new Selenium tutorials.

****

Here we are listing all the Selenium Training Tutorials for your handy reference.

List of Selenium Online Training Tutorials:

****

Selenium Basic

Selenium Basics:

- Tutorial #1 – Selenium Testing Introduction (Must Read)
- Tutorial #2 – Selenium IDE Features, Selenium Download and installation
- Tutorial #3 – My first Selenium IDE script (Must Read)
- Tutorial #4 – Creating script using Firebug and its installation
- Tutorial #5 – Locator Types: ID, ClassName, Name, Link Text, Xpath
- Tutorial #6 – Locator Types: CSS Selector
- Tutorial #7 – Locating elements in Google Chrome and IE

Selenium Training Tutorials

After hundreds of requests from STH readers, today we are finally ***launching our FREE Selenium Tutorial series***. In this Selenium training series we will cover all Selenium testing concepts and its packages in detail with easy to understand practical examples.

These Selenium tutorials are helpful for beginner to advanced level Selenium users. Starting from the very basic Selenium concepts tutorial, we will gradually move on to the advanced topics like Framework creation, Selenium Grid and Cucumber BDD.

Note*: We will be increasing our article posting frequency for this series. Please don't miss any tutorial. Keep track of all the tutorials by bookmarking this page as we will keep updating it with links to all new Selenium tutorials.*

How to start Learning Selenium?

This is the best time to start learning Selenium testing by your own with the help of this free Selenium Training series. Read tutorials, practice examples at your home, and put your queries in comment section of respective tutorials. We will address all of these queries.

Experienced Selenium professionals – you too can take part in this series by providing answers to reader's queries in comments.

This is our serious effort to help you learn and master one of the most popular software testing tools!

Selenium Introduction:

We are delighted to launch our yet another series of software testing training tutorials. The belief behind introducing this tutorial is to make you an expert in a widely used software test automation solution, Selenium. In this series we will look at the various facets of Selenium. Selenium is not just a tool; it is a cluster of independent tools. We will look into some of the tools in detail, providing practical examples wherever applicable.

Before you jump in to reading this exciting and useful series, let us take a look at what it has got in store for you.

Why Selenium?

As the current industry trends have shown that there is mass movement towards automation testing. The cluster of repetitive manual testing scenarios has raised a demand to bring in the practice of automating these manual scenarios.

The benefits of implementing automation test are many; let us take a look at them:

- Supports execution of repeated test cases
- Aids in testing a large test matrix
- Enables parallel execution
- Encourages unattended execution
- Improves accuracy thereby reducing human generated errors
- Saves time and money

All this results in to the following:

- High ROI
- Faster GoTo market

Automation testing benefits are many and well understood and largely talked about in the software test industry.

One of the most commonly asked question comes with this is –

- What is the best tool for me to get my tests automated?
- Is there a cost involved?
- Is it easy to adapt?

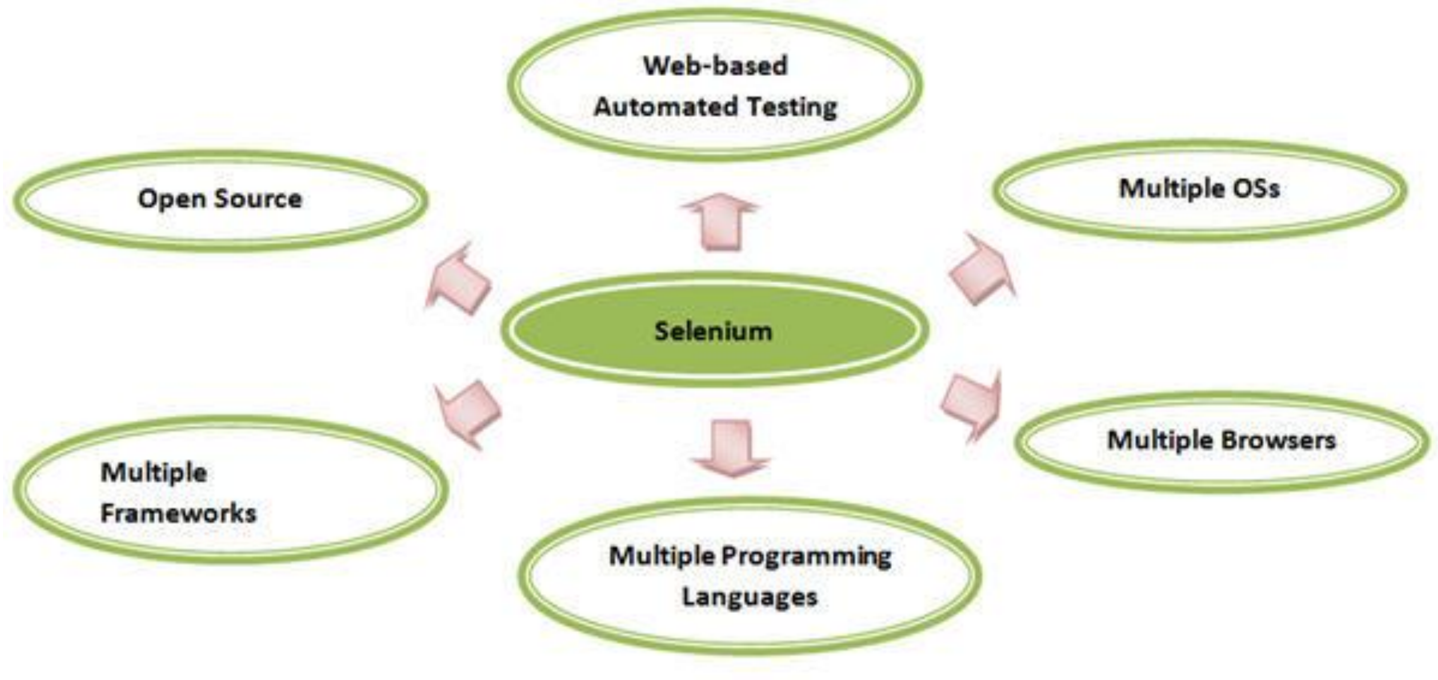
One of the best answers to all the above questions for automating web based applications is Selenium. Because:

- It's open source
- have a large user base and helping communities
- have multi browser and platform compatibility
- has active repository developments
- supports multiple language implementations

First glance at Selenium

Selenium is one of the most popular automated testing suites. Selenium is designed in a way to support and encourage automation testing of functional aspects of web based applications and a wide range of browsers and platforms. Due to its existence in the open source community, it has become one of the most accepted tools amongst the testing professionals.

Selenium supports a broad range of browsers, technologies and platforms.



Selenium supports a broad range of browsers, technologies and platforms

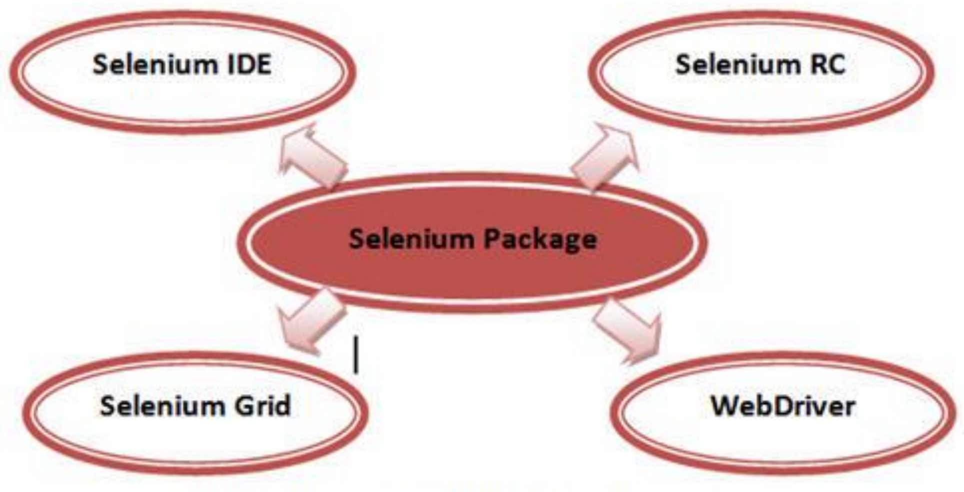
Selenium Components

Selenium is not just a single tool or a utility, rather a package of several testing tools and for the same reason it is referred to as a Suite. Each of these tools is designed to cater different testing and test environment requirements.

The suite package constitutes of the following sets of tools:

- Selenium Integrated Development Environment (IDE)
- Selenium Remote Control (RC)
- Selenium WebDriver
- Selenium Grid

Selenium RC and WebDriver, in a combination are popularly known as Selenium 2. Selenium RC alone is also referred as Selenium 1.



Selenium Packages - © www.SoftwareTestingHelp.com

Brief Introduction to Selenium tools

Selenium Core

Selenium is a result of continuous efforts by an engineer at ThoughtWorks, named as Jason Huggins. Being responsible for the testing of an internal Time and Expenses application, he realized the need for an automation testing tool so as to get rid of repetitive manual tasks without compromising with the quality and accuracy.

As a result, he built a JavaScript program, named as "JavaScriptTestRunner" in early 2004 that could automatically control the browser's actions which seemed very much similar to that of a user communicating with the browser.

Henceforth, Jason started demoing the tool to the vast audience. Eventually the discussions were laid out to categorize this tool in the open source category as well as its potential to grow as a re-usable testing framework for other web based applications.

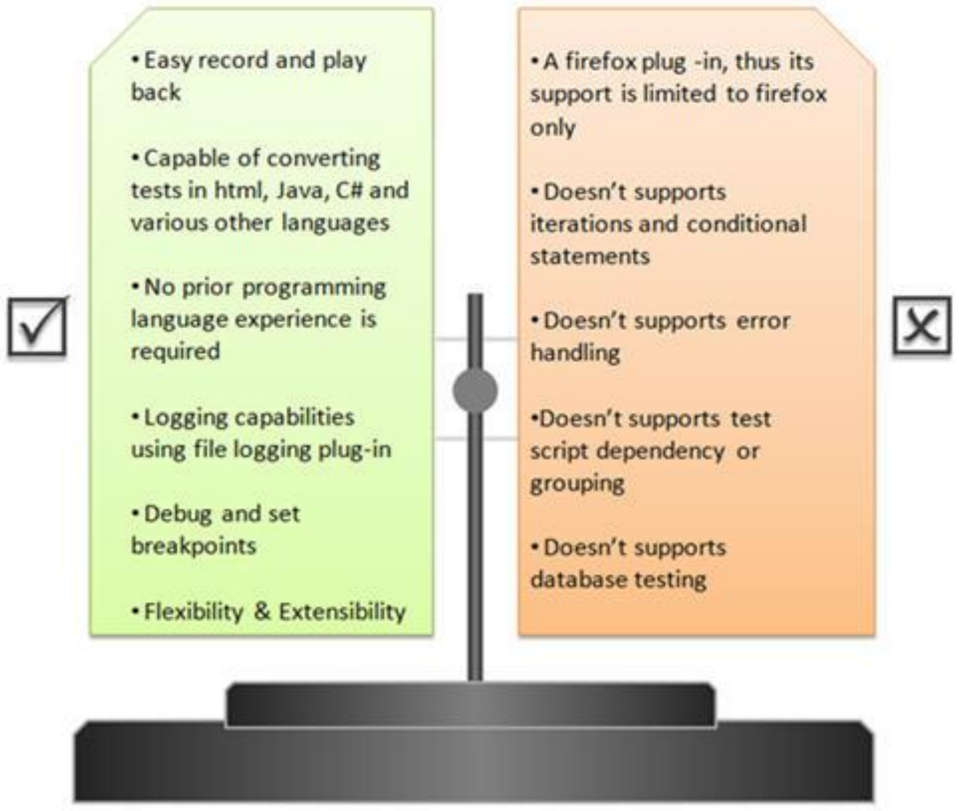
The tool was later on acclaimed with the name "Selenium Core".

Selenium IDE (Selenium Integrated Development Environment)

Selenium IDE was developed by Shinya Kasatani. While studying Selenium Core, he realized that this JavaScript code can be extended to create an integrated development environment (IDE) which can be plugged into Mozilla Firefox. This IDE was capable of recording and playing back the user actions on a Firefox instance to which it was plugged-in. Later on Selenium IDE became a part of Selenium Package in the year 2006. The tool turned out a great value and potential to the community.

Selenium IDE is the simplest and easiest of all the tools within the Selenium Package. Its record and playback feature makes it exceptionally easy to learn with minimal acquaintances to any programming language. With several advantages, a few disadvantages accompanied Selenium IDE, thus making it inappropriate to be used in cases of more advanced test scripts.

Advantages and disadvantages of Selenium IDE:


•Easy record and play back
•Capable of converting tests in html, Java, C# and various other languages
•No prior programming language experience is required
•Logging capabilities using file logging plug-in
•Debug and set breakpoints
•Flexibility & Extensibility
•A firefox plug -in, thus its support is limited to firefox only
•Doesn't supports iterations and conditional statements
•Doesn't supports error handling
•Doesn't supports test script dependency or grouping
•Doesn't supports database testing

Advantages and disadvantages of Selenium IDE - © www.SoftwareTestingHelp.com

The disadvantages of IDE are in reality not disadvantages of selenium, rather just limitations to what IDE could achieve. These limitations can be overcome by using Selenium RC or WebDriver.

Selenium RC (Selenium Remote Control)

Selenium RC is a tool which is written in java that allows a user to construct test scripts for a web based application in which ever programming language he/she chooses. Selenium RC came as result to overcome various disadvantages incurred by Selenium IDE or Core.

Loopholes and restrictions which were imposed while using Selenium Core made it difficult for the user to leverage the benefits of the tool to its totality. Thus it made the testing process a cumbersome and a far reaching task.

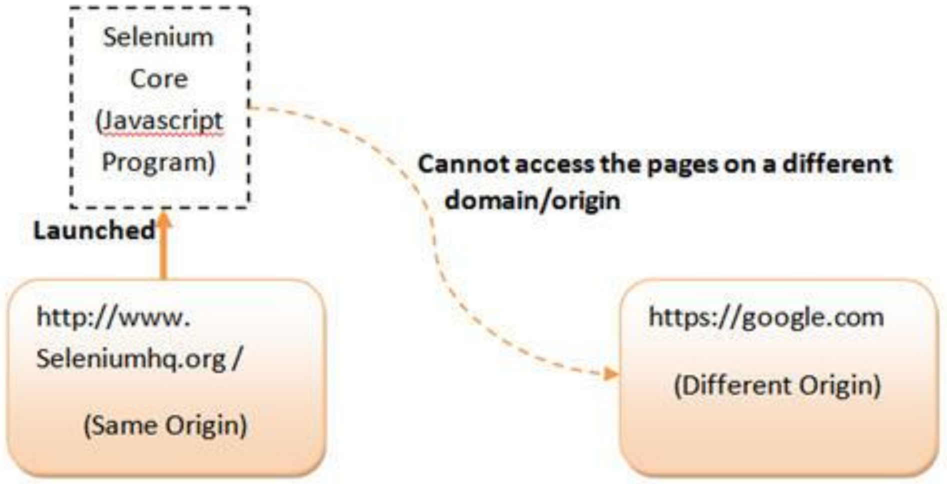
One of the crucial restrictions was **same origin policy.**

Problem of same origin policy:

The problem of same origin policy disallows to access the DOM of a document from an origin that is different from the origin we are trying to access the document.

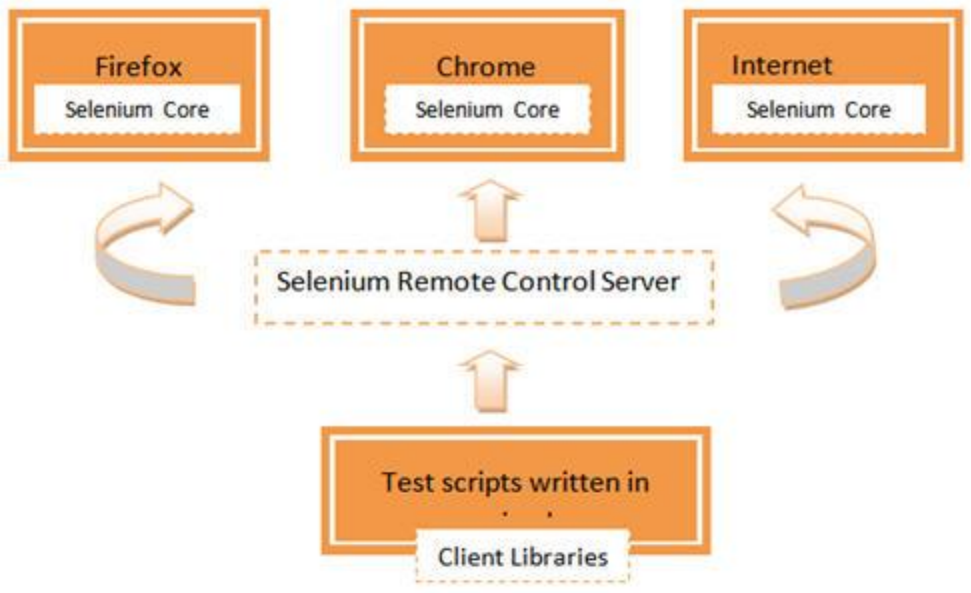
Origin is a sequential combination of scheme, host and port of the URL. For example, for a URL http://www.seleniumhq.org/projects/, the origin is a combination of http, seleniumhq.org, 80 correspondingly. Thus the Selenium Core (JavaScript Program) cannot access the elements from an origin that is different from where it was launched.

For Example, if I have launched the JavaScript Program from "http://www.seleniumhq.org/", then I would be able to access the pages within the same domain such as "http://www.seleniumhq.org/projects/" or "http://www.seleniumhq.org/download/". The other domains like google.com, yahoo.com would no more be accessible.
Thus, to test the application using Selenium Core, one has to install the entire application on the Selenium Core as well as web server to overcome the problem of same origin policy.


Selenium Core (Javascript Program)
Cannot access the pages on a different domain/origin
Launched
http://www. Seleniumhq.org /
(Same Origin)
https://google.com
(Different Origin)

So, In order to govern the same origin policy without the need of making a separate copy of Application under test on the Selenium Core, Selenium Remote Control was introduced. While Jason Huggins was demoing Selenium, another fellow colleague at ThoughtWorks named Paul Hammant suggested a work around of same origin policy and a tool that can be wired up with a programming language of our choice. Thus Selenium RC came into existence.

Unlike selenium IDE, selenium RC supports a wide range of browsers and platforms.



Selenium RC Supports - © www.SoftwareTestingHelp.com

Workflow Description

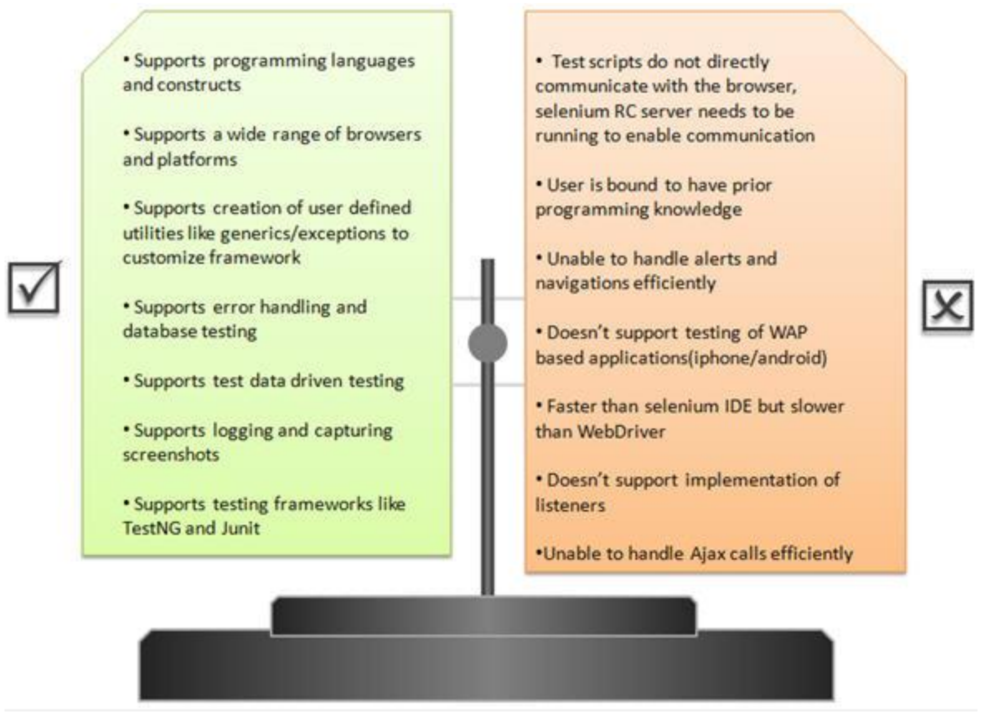
- User creates test scripts in a desired programming language.
- For every programming language, there is a designated client library.
- Client library deports the test commands to the selenium server.
- Selenium server deciphers and converts the test commands into JavaScript commands and sends them to the browser.
- Browser executes the commands using selenium core and sends results back to the selenium server
- Selenium server delivers the test results to the client library.

There are a few pre-requisites to be in place before creating Selenium RC scripts:

- A Programming Language – Java, C#, Python etc.
- An Integrated Development Environment –Eclipse, Netbeans etc.
- A Testing Framework (optional) – JUnit, TestNG etc.
- And Selenium RC setup off course

Advantages and disadvantages of selenium RC:

Coming on to the advantages and disadvantages of selenium RC, refer the following figure.



• Supports programming languages and constructs
• Supports a wide range of browsers and platforms
• Supports creation of user defined utilities like generics/exceptions to customize framework
• Supports error handling and database testing
• Supports test data driven testing
• Supports logging and capturing screenshots
• Supports testing frameworks like TestNG and Junit
• Test scripts do not directly communicate with the browser, selenium RC server needs to be running to enable communication
• User is bound to have prior programming knowledge
• Unable to handle alerts and navigations efficiently
• Doesn't support testing of WAP based applications(iphone/android)
• Faster than selenium IDE but slower than WebDriver
• Doesn't support implementation of listeners
•Unable to handle Ajax calls efficiently

Advantages and disadvantages of Selenium RC - © www.SoftwareTestingHelp.com

Selenium Grid

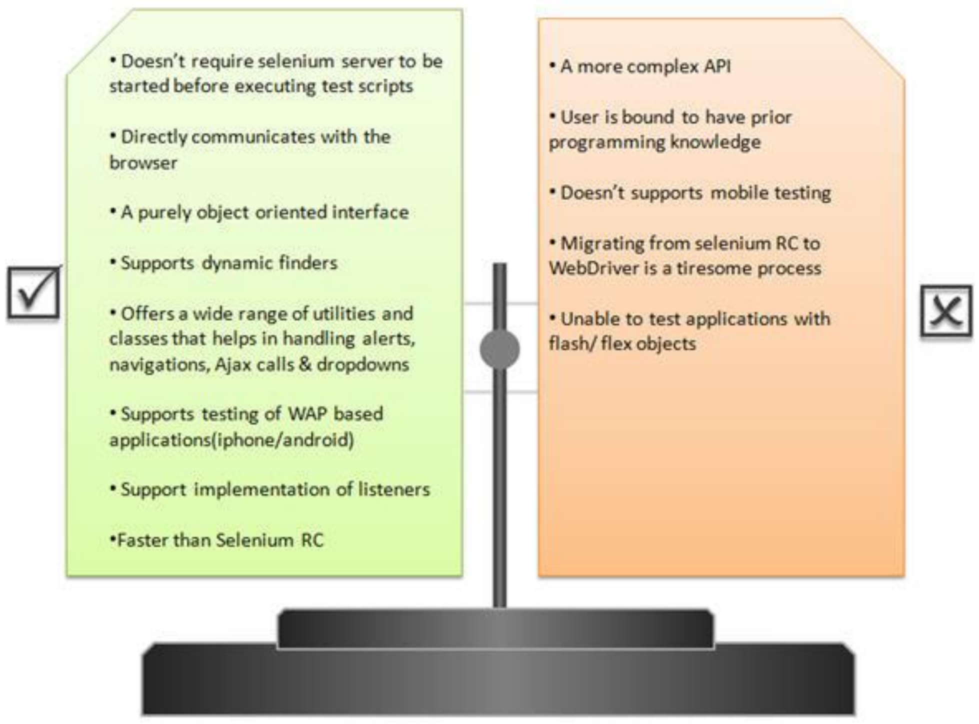
With selenium RC, life of a tester has always been positive and favorable until the emerging trends raised a demand to execute same or different test scripts on multiple platforms and browsers concurrently so as to achieve distributed test execution, testing under different environments and saving execution time remarkably. Thus, catering these requirements selenium grid was brought into the picture.

Selenium Grid was introduced by Pat Lightbody in order to address the need for executing the test suites on multiple platforms simultaneously.

Selenium WebDriver

Selenium WebDriver was created by yet another engineer at ThoughtWorks named as Simon Stewart in the year 2006. WebDriver is also a web-based testing tool with a subtle difference with Selenium RC. Since, the tool was built on the fundamental where an isolated client was created for each of the web browser; no JavaScript Heavy lifting was required. This led to a compatibility analysis between Selenium RC and WebDriver. As a result a more powerful automated testing tool was developed called ***Selenium 2***.
WebDriver is clean and a purely object oriented framework. It utilizes the browser's native compatibility to automation without using any peripheral entity. With the increasing demand it has gained a large popularity and user base.

Advantages and disadvantages of Selenium WebDriver:
Refer the following figure for the advantages and disadvantages of WebDriver.



• Doesn't require selenium server to be started before executing test scripts
• Directly communicates with the browser
• A purely object oriented interface
• Supports dynamic finders
• Offers a wide range of utilities and classes that helps in handling alerts, navigations, Ajax calls & dropdowns
• Supports testing of WAP based applications(iphone/android)
• Support implementation of listeners
•Faster than Selenium RC
• A more complex API
• User is bound to have prior programming knowledge
• Doesn't supports mobile testing
• Migrating from selenium RC to WebDriver is a tiresome process
• Unable to test applications with flash/ flex objects

Advantages and disadvantages of Selenium WebDriver - © www.SoftwareTestingHelp.com

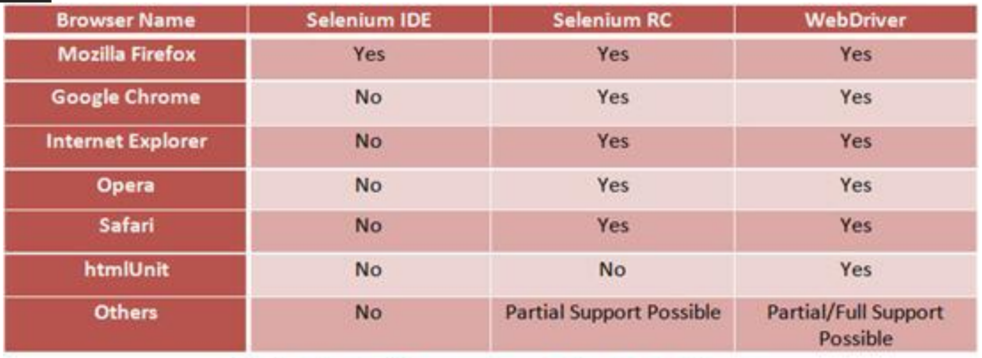
Selenium 3

Selenium 3 is an advance version of Selenium 2. It is a tool focused for automation of mobile and web applications. Stating that it supports mobile testing, we mean to say that the WebDriver API has been extended to address the needs of mobile application testing. The tool is expected to be launched soon in the market.

Environment and Technology Stack

With the advent and addition of each new tool in the selenium suite, environments and technologies became more compatible. Here is an exhaustive list of environments and technologies supported by selenium tool set.

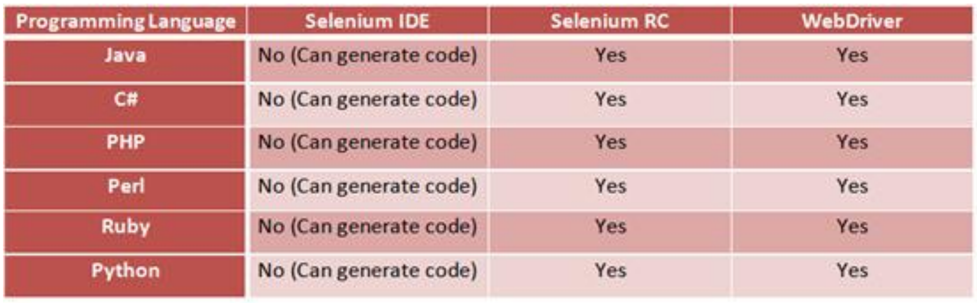
Supported Browsers



Browser Name	Selenium IDE	Selenium RC	WebDriver
Mozilla Firefox	Yes	Yes	Yes
Google Chrome	No	Yes	Yes
Internet Explorer	No	Yes	Yes
Opera	No	Yes	Yes
Safari	No	Yes	Yes
htmlUnit	No	No	Yes
Others	No	Partial Support Possible	Partial/Full Support Possible

Selenium Supported Browsers - © www.SoftwareTestingHelp.com

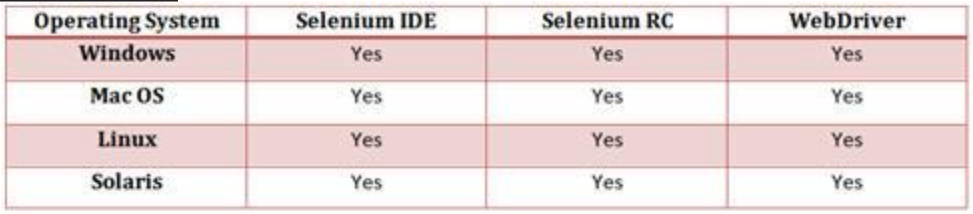
Supported Programming Languages



Programming Language	Selenium IDE	Selenium RC	WebDriver
Java	No (Can generate code)	Yes	Yes
C#	No (Can generate code)	Yes	Yes
PHP	No (Can generate code)	Yes	Yes
Perl	No (Can generate code)	Yes	Yes
Ruby	No (Can generate code)	Yes	Yes
Python	No (Can generate code)	Yes	Yes

Selenium Supported Languages - © www.SoftwareTestingHelp.com

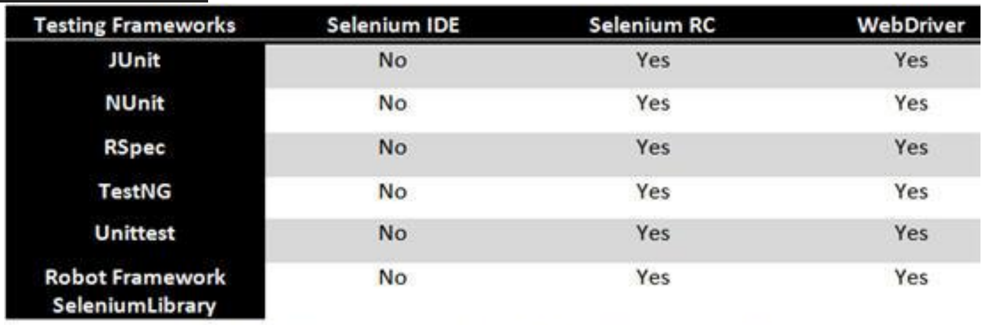
<u>**Supported Operating Systems**</u>



Operating System	Selenium IDE	Selenium RC	WebDriver
Windows	Yes	Yes	Yes
Mac OS	Yes	Yes	Yes
Linux	Yes	Yes	Yes
Solaris	Yes	Yes	Yes

Selenium Supported OSes - © www.SoftwareTestingHelp.com

<u>**Supported Testing Frameworks**</u>



Testing Frameworks	Selenium IDE	Selenium RC	WebDriver
JUnit	No	Yes	Yes
NUnit	No	Yes	Yes
RSpec	No	Yes	Yes
TestNG	No	Yes	Yes
Unittest	No	Yes	Yes
Robot Framework SeleniumLibrary	No	Yes	Yes

Selenium Supported Frameworks - © www.SoftwareTestingHelp.com

Conclusion

In this tutorial, we tried to make you acquainted with the Selenium suite describing its various components, their usages and their advantages over each other.

Here are the cruxes of this article.

- Selenium is a suite of several automated testing tools, each of them catering to different testing needs.
- All these tools fall under the same umbrella of open source category and supports only web based testing.
- Selenium suite is comprised of 4 basic components; Selenium IDE, Selenium RC, WebDriver, Selenium Grid.
- User is expected to choose wisely the right Selenium tool for his/her needs.
- Selenium IDE is distributed as a Firefox plug-in. It is easier to install and use. User is not required to possess prior programming knowledge. Selenium IDE is an ideal tool for a naive user.
- Selenium RC is a server that allows user to create test scripts in a desired programming language. It also allows executing test scripts within the large spectrum of browsers.

- Selenium Grid brings out an additional feature to Selenium RC by distributing its test script on different platforms and browsers at the same time for execution, thus implementing the master slave architecture.
- WebDriver is a different tool altogether that has various advantages over Selenium RC. The fusion of Selenium RC and WebDriver is also known as Selenium 2. WebDriver directly communicates with the web browser and uses its native compatibility to automate.
- Selenium 3 is the most anticipated inclusion in the Selenium suite which is yet to be launched in the market. Selenium 3 strongly encourages mobile testing.

In the next tutorial, we would be discussing about the basics of Selenium IDE, its installation and its features. We would also have a look at the basic terminologies and nomenclatures of Selenium IDE.

***Next Selenium Tutorial*: *Introduction to Selenium IDE and its installation* with detailed study on all the features of Selenium IDE (coming soon)**

A remark for the readers: While our next tutorial of the Selenium training series is in the processing mode, meanwhile you can explore a bit about the Selenium suite and its tools by looking at its official website.

About the authors:

Shruti Shrivastava (our main author for this series), Amaresh Dhal, and Pallavi Sharma are helping us to bring this series to our readers.

Shruti is currently working as a Senior Test Engineer with 4+ years of automation testing experience. She is an ISTQB certified professional and also an active blogger, always interested in solving testing related problems.

Amaresh is having 5+ years of manual and automation testing experience with expertise in WebDriver, Grid and frameworks.

Pallavi Sharma has more than 7 years rich experience of working in automation testing field with hands-on Selenium and JAVA experience.

Stay tuned till then and share your views, comments and knowledge to help us groom. Also let us know if you find anything that we missed out so that we can include them in the subsequent tutorials.

Getting Started with Selenium IDE (Installation and its Features) – Selenium Tutorial #2

Before moving ahead, let's take a moment to look at the agenda of this tutorial. In this tutorial, we will learn all about ***Selenium IDE***, starting from its installation to the details about each of its features. At the end of this tutorial, the reader is expected to be able to install Selenium IDE tool and play around with its features.
=> This is a 2nd tutorial in our free online Selenium training series. If you have not read the first Selenium tutorial in this series please get started from here: **Free online Selenium Tutorial #1**
Note: *This is quite a extensive tutorial with lots of images so allow it to load completely. Also click on image or open in new window to enlarge images.*

Introduction to Selenium IDE

Selenium integrated development environment, acronym as Selenium IDE is an automated testing tool that is released as a Firefox plug-in. It is one of the simplest and easiest tools to install, learn and to go ahead with the creation of test scripts. The tool is laid on a record and playback fundamental and also allows editing of the recorded scripts.

The most impressive aspect of using selenium IDE is that the user is not required to possess any prior programming knowledge. The minimum that the user needs is the little acquaintances with HTML, DOMS and JavaScript to create numerous test scripts using this tool.

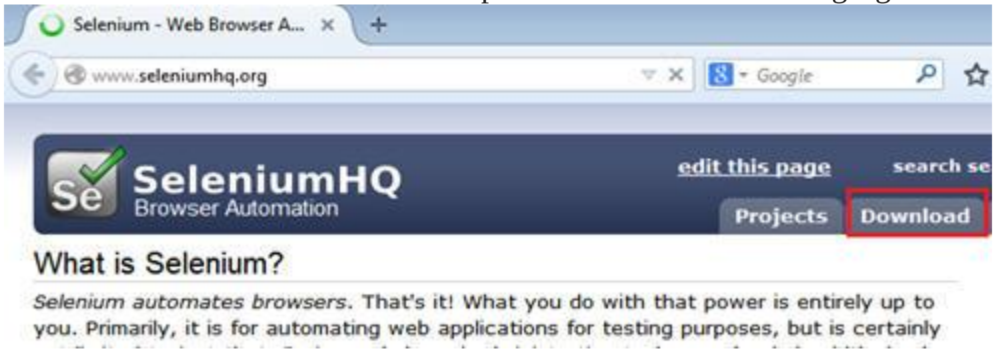
Being a Firefox plug-in, Selenium IDE supports only Firefox, thus the created test scripts could be executed only on Firefox. A few more loopholes make this tool inappropriate to be used for complex test scripts. Thus, other tools like Selenium RC, WebDriver comes into the picture.
So, before gripping on to the details of Selenium IDE, let's have a look at its installation first.

Selenium IDE Download and Installation

For the ease of understanding, I have bifurcated the entire IDE installation process in the following chunks/steps.

Before taking off, there is one thing that needs to be in place prior to the installation; Mozilla Firefox. You can download it from here => Mozilla Firefox download.

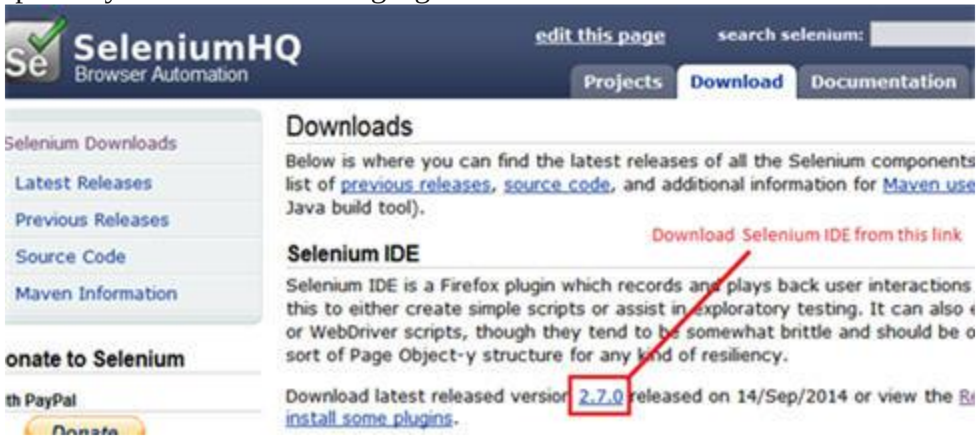
Step #1: Selenium IDE download: Open the browser (Firefox) and enter the URL – http://seleniumhq.org/ . This would open the official Selenium head quarter website. Navigate to the “Download” page; this page embodies all the latest releases of all the selenium components. Refer the following figure.



What is Selenium?

Selenium automates browsers. That's it! What you do with that power is entirely up to you. Primarily, it is for automating web applications for testing purposes, but is certainly

Step #2: Move under the selenium IDE head and click on the link present. This link represents the latest version of the tool in the repository. Refer the following figure.



Downloads

Below is where you can find the latest releases of all the Selenium components list of previous releases, source code, and additional information for Maven use Java build tool).

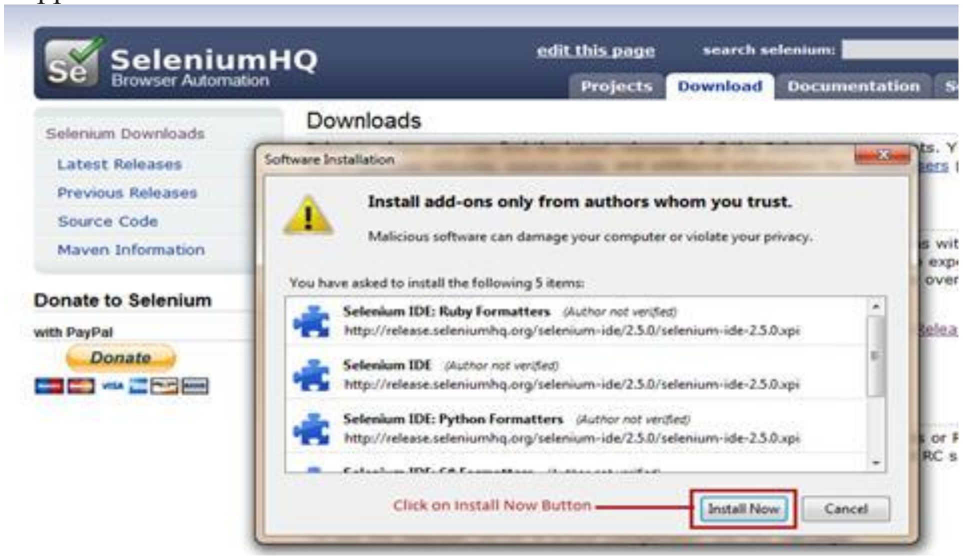
Selenium IDE

Selenium IDE is a Firefox plugin which records and plays back user interactions this to either create simple scripts or assist in exploratory testing. It can also or WebDriver scripts, though they tend to be somewhat brittle and should be o sort of Page Object-y structure for any kind of resiliency.

Download latest released version 2.7.0 released on 14/Sep/2014 or view the R install some plugins.

Step #3: As soon as we click on the above link, a security alert box would appear so as to safeguard our system against potential risks. As we are downloading the plug-in from the authentic website, thus click on the “Allow” button.

Step #4: Now Firefox downloads the plug-in in the backdrop. As soon as the process completes, software installation window appears. Now click on the “Install Now” button.



Install add-ons only from authors whom you trust.

Malicious software can damage your computer or violate your privacy.

You have asked to install the following 5 items:

Selenium IDE: Ruby Formatters (Author not verified)
http://release.seleniumhq.org/selenium-ide/2.5.0/selenium-ide-2.5.0.xpi

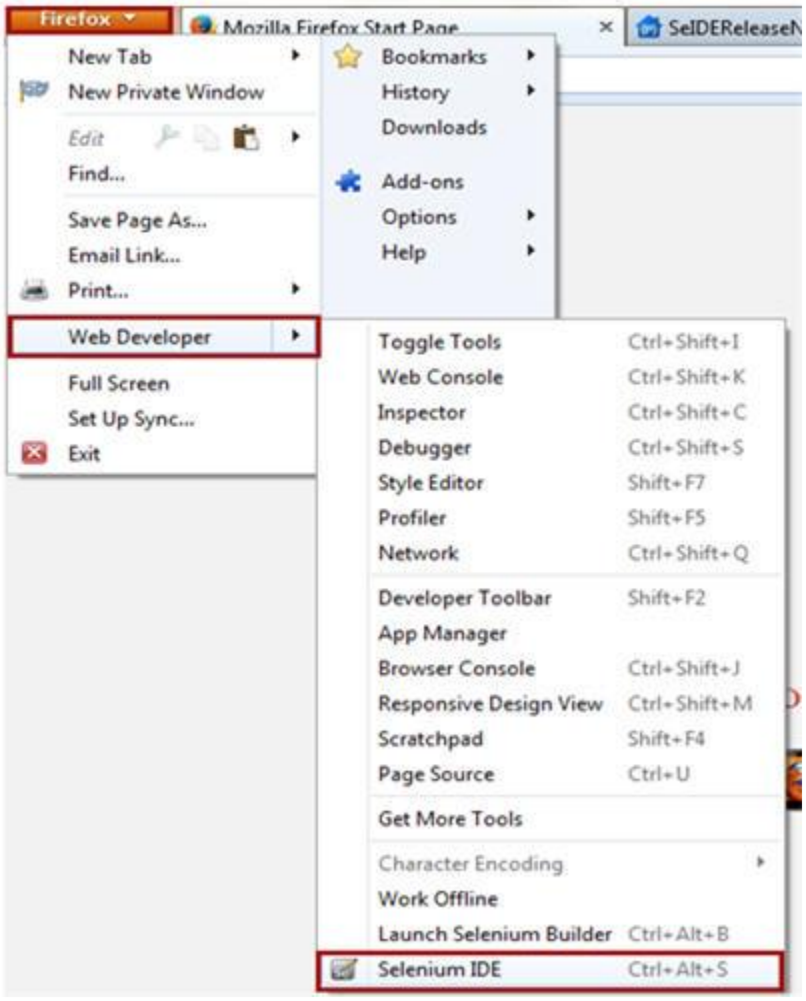
Selenium IDE (Author not verified)
http://release.seleniumhq.org/selenium-ide/2.5.0/selenium-ide-2.5.0.xpi

Selenium IDE: Python Formatters (Author not verified)
http://release.seleniumhq.org/selenium-ide/2.5.0/selenium-ide-2.5.0.xpi

Selenium IDE: C# Formatters (Author not verified)

Step #5: After the installation is completed, a pop up window appears asking to re-start the Firefox. Click on the "Restart Now" button to reflect the Selenium IDE installation.

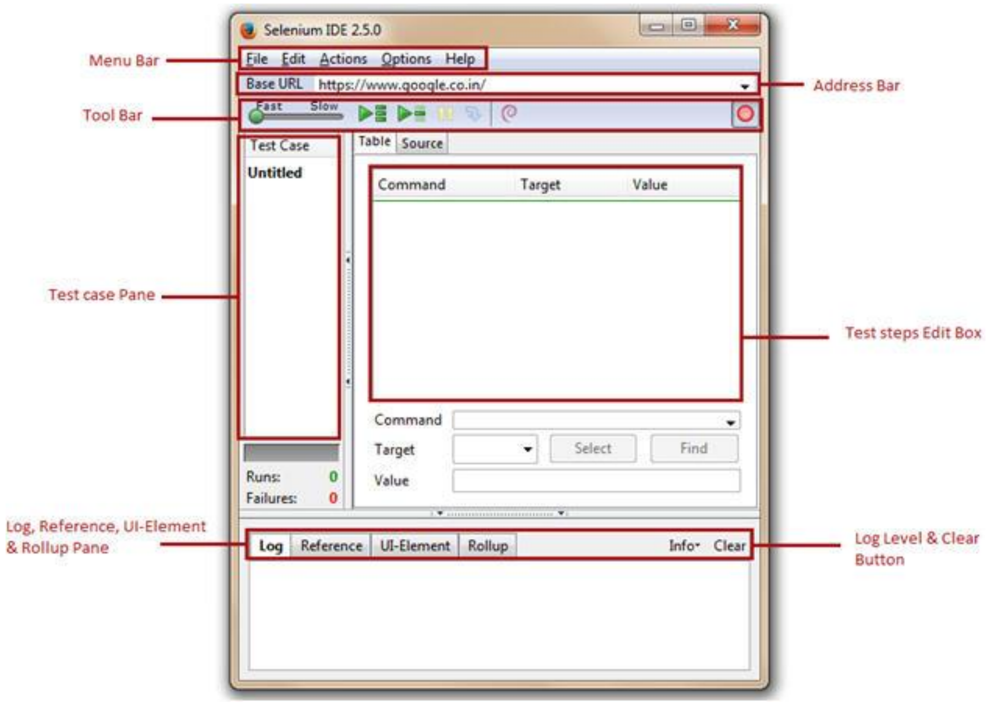
Step #6: Once the Firefox is booted and started again, we can see selenium IDE indexed under menu bar -> Web Developer -> Selenium IDE.



Step #7: As soon as we open Selenium IDE, the Selenium IDE window appears.

<u>Features of Selenium IDE</u>

Let's have a look at each of the feature in detail.

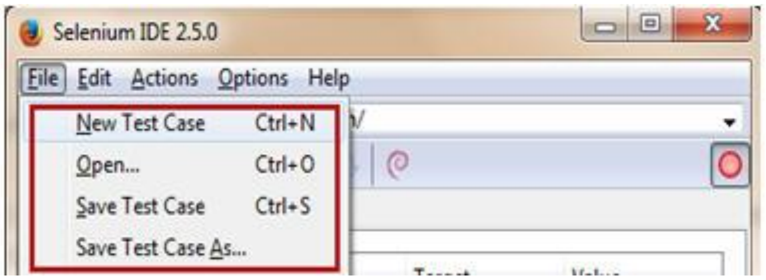


#1. Menu Bar

Menu bar is positioned at the upper most of the Selenium IDE window. The menu bar is typically comprised of five modules.

- File Menu
- Edit Menu
- Actions Menu
- Options Menu
- Help Menu

A) File Menu

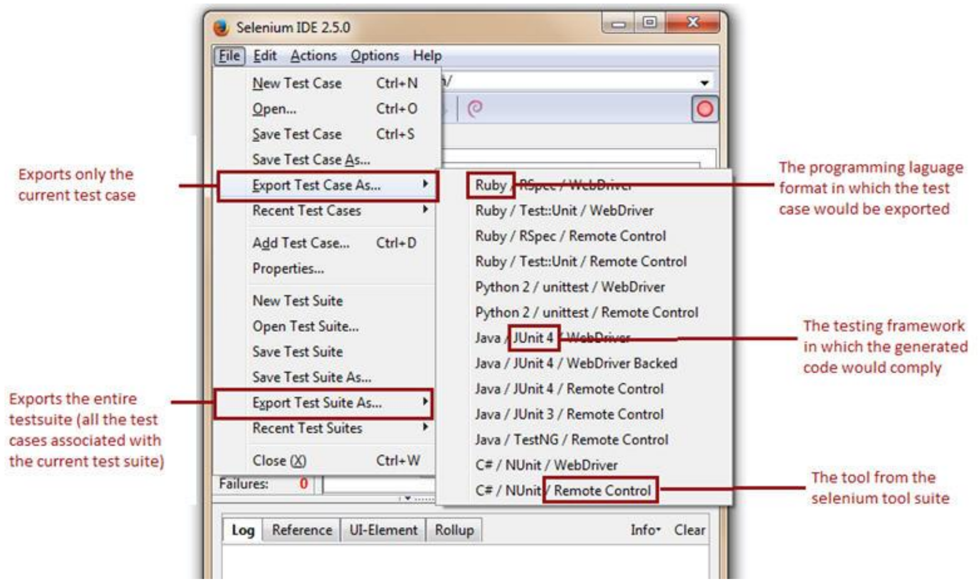


File Menu is very much analogous to the file menu belonging to any other application. It allows user to:

- Create new test case, open existing test case, save the current test case.
- Export Test Case As and Export Test Suite As in any of the associated programming language compatible with Selenium RC and WebDriver. It also gives the liberty to the user to prefer amid the

available unit testing frameworks like jUnit, TestNG etc. Thus an IDE test case can be exported for a chosen union of programming language, unit testing framework and tool from the selenium package.

- Export Test Case As option exports and converts only the currently opened Selenium IDE test case.
- Export Test Suite As option exports and converts all the test cases associated with the currently opened IDE test suite.
- Close the test case.

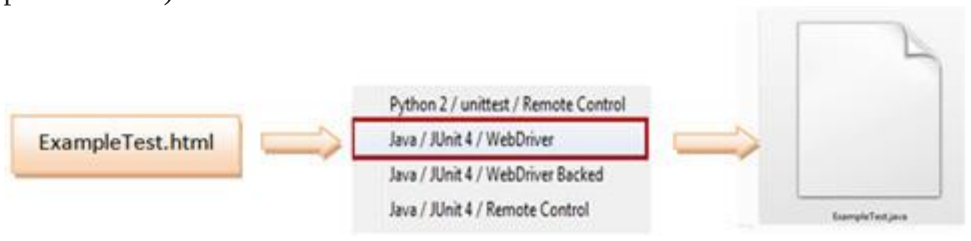

Selenium IDE 2.5.0
File Edit Actions Options Help
New Test Case Ctrl+N
Open... Ctrl+O
Save Test Case Ctrl+S
Save Test Case As...
Export Test Case As...
Recent Test Cases
Add Test Case... Ctrl+D
Properties...
New Test Suite
Open Test Suite...
Save Test Suite
Save Test Suite As...
Export Test Suite As...
Recent Test Suites
Close (X) Ctrl+W
Failures: 0
Log Reference UI-Element Rollup Info Clear
Ruby / RSpec / WebDriver
Ruby / Test::Unit / WebDriver
Ruby / RSpec / Remote Control
Ruby / Test::Unit / Remote Control
Python 2 / unittest / WebDriver
Python 2 / unittest / Remote Control
Java / JUnit 4 / WebDriver
Java / JUnit 4 / WebDriver Backed
Java / JUnit 4 / Remote Control
Java / JUnit 3 / Remote Control
Java / TestNG / Remote Control
C# / NUnit / WebDriver
C# / NUnit / Remote Control
Exports only the current test case
Exports the entire testsuite (all the test cases associated with the current test suite)
The programming laguage format in which the test case would be exported
The testing framework in which the generated code would comply
The tool from the selenium tool suite

The Selenium IDE test cases can be saved into following format:

- HTML format

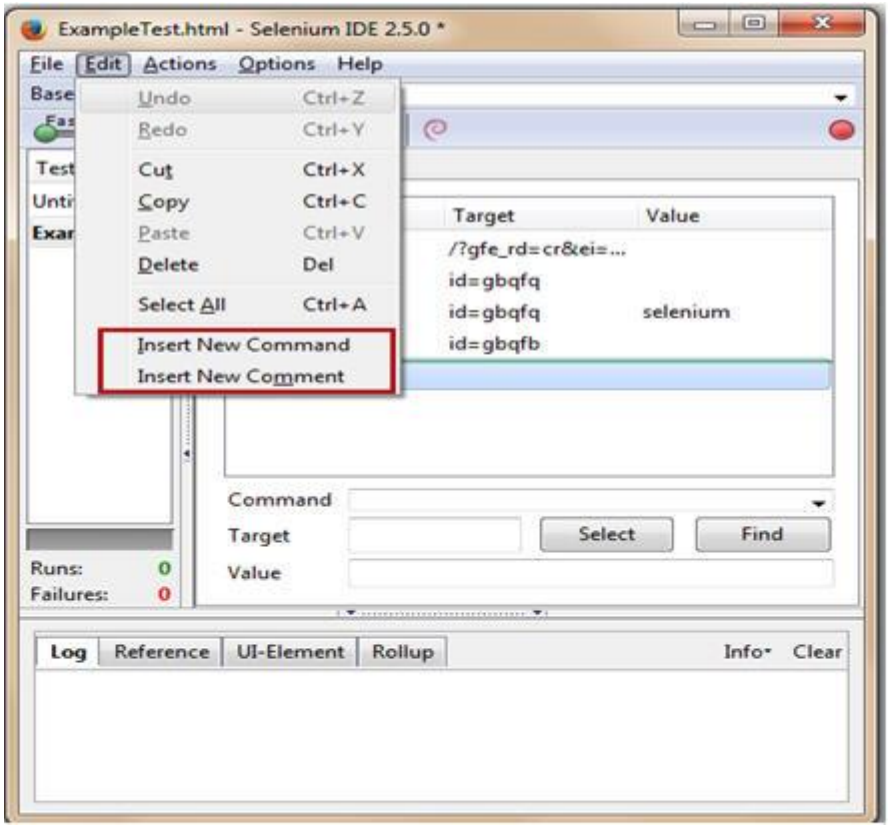
The Selenium IDE test cases can be exported into following formats/programming languages.

- java (IDE exported in Java)
- rb (IDE exported in Ruby)
- py (IDE exported in Python)
- cs (IDE exported in C#)


ExampleTest.html
Python 2 / unittest / Remote Control
Java / JUnit 4 / WebDriver
Java / JUnit 4 / WebDriver Backed
Java / JUnit 4 / Remote Control
ExampleTest.java

Notice that with the forthcoming newer versions of Selenium IDE, the support to formats may expand.

B) Edit Menu

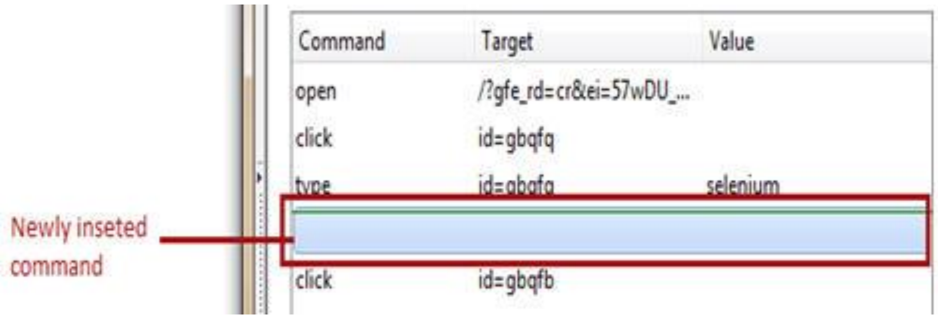


Edit menu provides options like Undo, Redo, Cut, Copy, Paste, Delete and Select All which are routinely present in any other edit menu. Amongst them, noteworthy are:

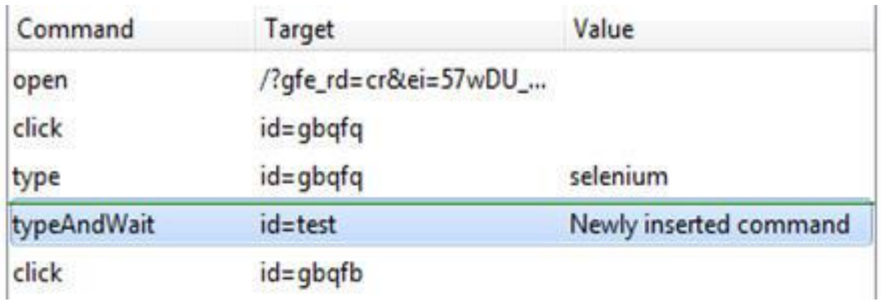
- Insert New Command – Allows user to insert the new command/test step anywhere within the current test case.
- Insert New Comment – Allows user to insert the new comment anywhere within the current test case to describe the subsequent test steps.

Insert New Command

The new command would be inserted above the selected command/test step.



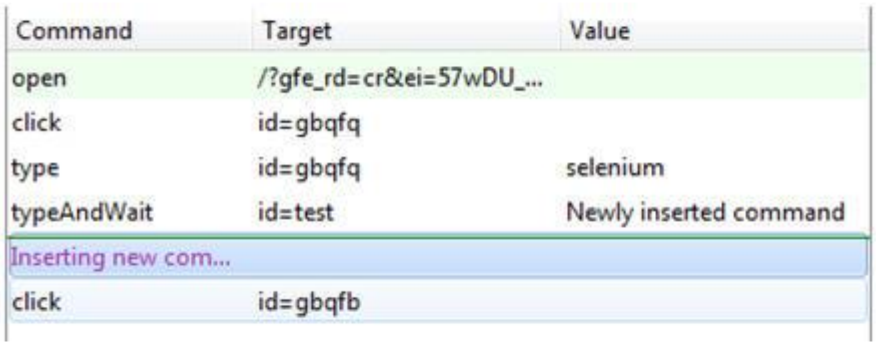
Now the user can insert the actual command action, target and value.



Command	Target	Value
open	/?gfe_rd=cr&ei=57wDU_...	
click	id=gbqfq	
type	id=gbqfq	selenium
typeAndWait	id=test	Newly inserted command
click	id=gbqfb	

Insert New Comment

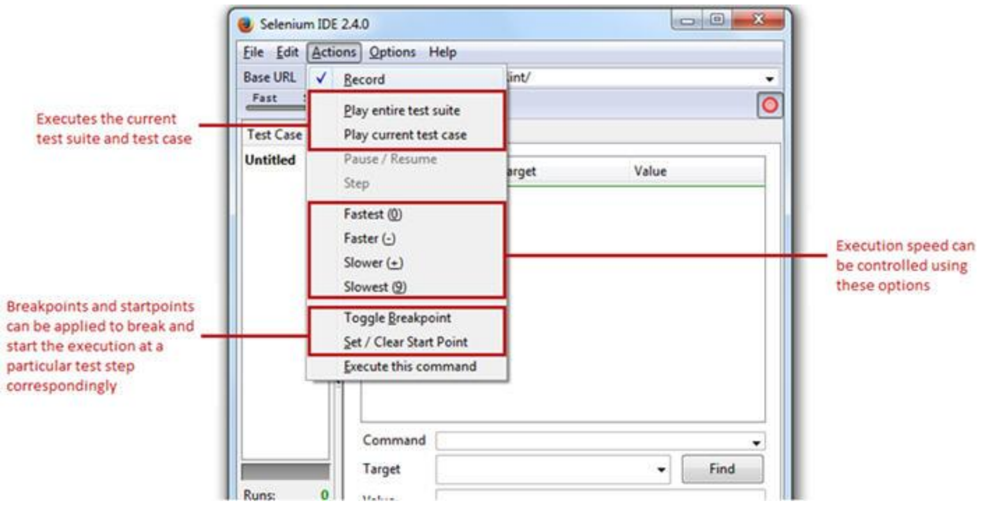
In the same way we can insert comments.



Command	Target	Value
open	/?gfe_rd=cr&ei=57wDU_...	
click	id=gbqfq	
type	id=gbqfq	selenium
typeAndWait	id=test	Newly inserted command
Inserting new com...		
click	id=gbqfb	

The purple color indicates that the text is representing a comment.

C) Actions Menu

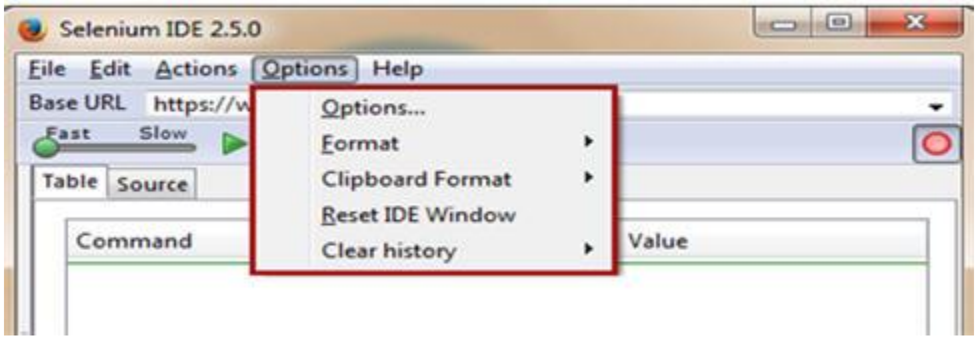


Actions Menu equips the user with the options like:

- **Record** – Record options fine tunes the Selenium IDE into the recording mode. Thus, any action made by the user on the Firefox browser would be recorded in IDE.
- **Play entire test suite** – The option plays all the Selenium IDE test cases associated with the current test suite.
- **Play current test case** – The option plays the current Selenium IDE test case that has been recorded/created by the user.

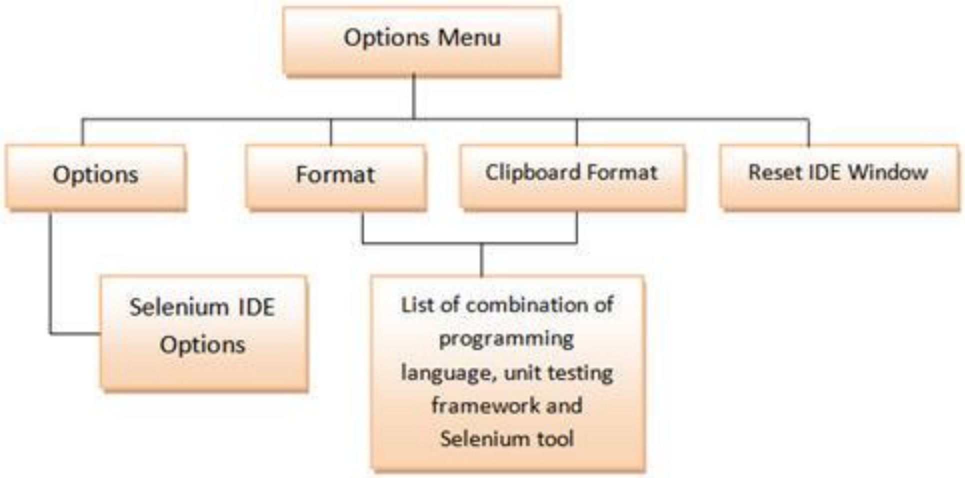
- **Pause / Resume –** User can Pause/Resume the test case at any point of time while execution.
- **Toggle Breakpoint** – User can set one or multiple breakpoint(s) to forcefully break the execution at any particular test step during execution.
- **Set / Clear Start Point** – User can also set start point at any particular test step for execution. This would enable user to execute the test case from the given start point for the subsequent runs.
- To deal with the page/element loads, the user can set the execution speed from fastest to lowest with respect to the responsiveness of the application under test.

D) Options Menu



Options menu privileges the user to set and practice various settings provided by the Selenium IDE. Options menu is recommended as one of the most important and advantageous menu of the tool.

Options Menu is primarily comprised of the following four components which can be sub-divided into the following:



Options Menu
Options
Format
Clipboard Format
Reset IDE Window
Selenium IDE Options
List of combination of programming language, unit testing framework and Selenium tool

Options

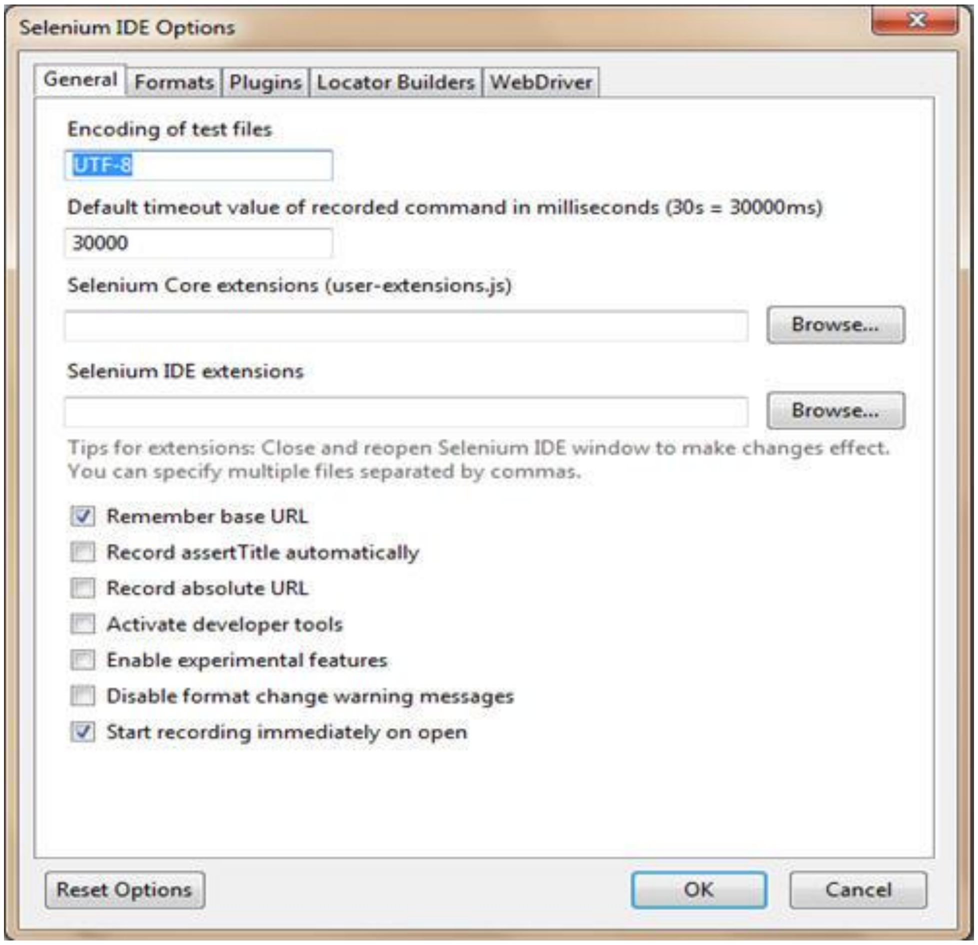
Selenium IDE Options dialog box

To launch Selenium IDE Options dialog box, follow the steps:

1. Click on Options Menu

2. Click on the Options

A Selenium IDE Options dialog box appears. Refer the following figure.



Encoding of test files: UTF-8

Default timeout value of recorded command in milliseconds (30s = 30000ms): 30000

Selenium Core extensions (user-extensions.js)

Selenium IDE extensions

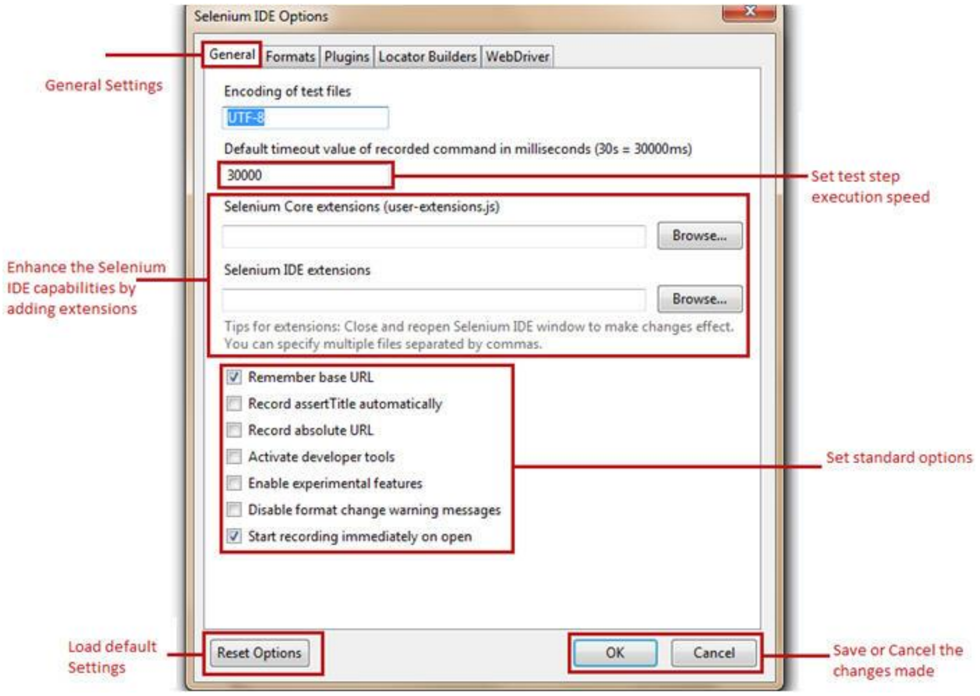
Tips for extensions: Close and reopen Selenium IDE window to make changes effect.
You can specify multiple files separated by commas.

- [✓] Remember base URL
- [ ] Record assertTitle automatically
- [ ] Record absolute URL
- [ ] Activate developer tools
- [ ] Enable experimental features
- [ ] Disable format change warning messages
- [✓] Start recording immediately on open

Selenium IDE Options dialog box aids the user to play with the general settings, available formats, available plug-ins and available locators types and their builders.

Let’s have a look at the few important ones.

General Settings



Encoding of test files: UTF-8

Default timeout value of recorded command in milliseconds (30s = 30000ms): 30000

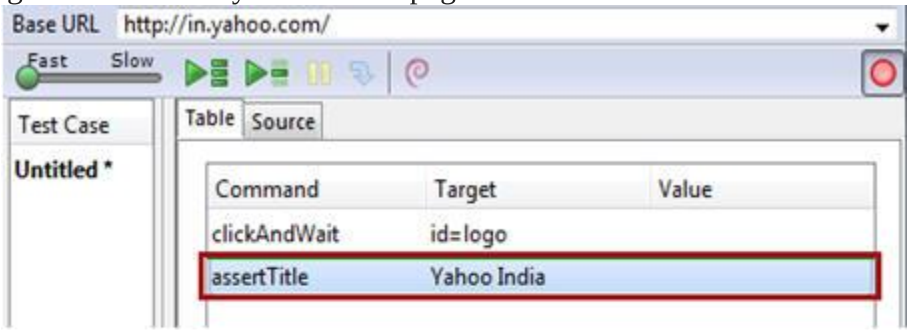
Selenium Core extensions (user-extensions.js)

Selenium IDE extensions

Tips for extensions: Close and reopen Selenium IDE window to make changes effect. You can specify multiple files separated by commas.

- [✓] Remember base URL
- [ ] Record assertTitle automatically
- [ ] Record absolute URL
- [ ] Activate developer tools
- [ ] Enable experimental features
- [ ] Disable format change warning messages
- [✓] Start recording immediately on open

- **Default Timeout Value** – Default Timeout Value represents the time (in milliseconds) that selenium would wait for a test step to execute before generating an error. The standard timeout value is 30000 milliseconds i.e. 30 seconds. The user can leverage this feature by changing the default time in cases when the web element takes more/less than the specified time to load.
- **Extensions** – Selenium IDE supports a wide range of extensions to enhance the capabilities of the core tool thereby multiplying its potential. These user extensions are simply the JavaScript files. They can set by mentioning their absolute path in the text boxes representing extensions in the Options dialog box.
- **Remember base URL** – Checking this option enables the Selenium IDE to remember the URL every time we launch it. Thus it is advisable to mark it checked. Un-checking this option will leave the base URL field as blank and it will be re-filled only when we launch another URL on the browser.
- **Record assertTitle automatically** – Checking this field inserts the assertTitle command automatically along with the target value for every visited web page.

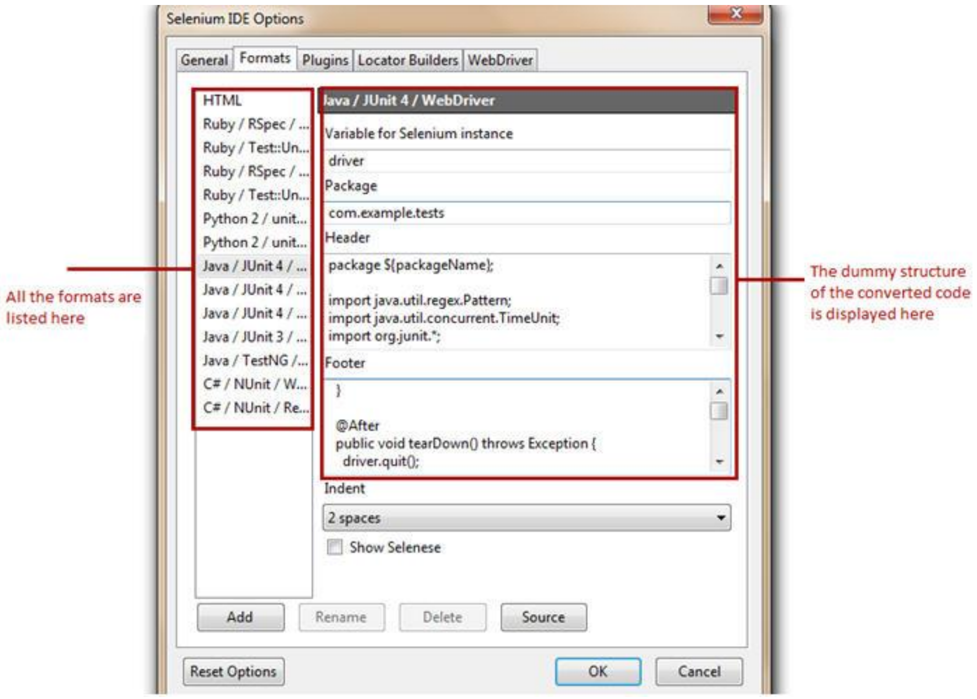
    - 

Base URL: http://in.yahoo.com/

Command	Target	Value
clickAndWait	id=logo	
assertTitle	Yahoo India	

- **Enable experimental features –** Checking this field for the first time imports the various available formats into the Selenium IDE.

Formats



Java / JUnit 4 / WebDriver

Variable for Selenium instance: driver

Package: com.example.tests

Header:

```
package ${packageName};

import java.util.regex.Pattern;
import java.util.concurrent.TimeUnit;
import org.junit.*;
```

Footer:

```
}

  @After
  public void tearDown() throws Exception {
    driver.quit();
```

Indent: 2 spaces

☐ Show Selenese

Formats tab displays all the available formats with selenium IDE. User is levied with the choice to enable and disable any of the formats. Refer the following figure.

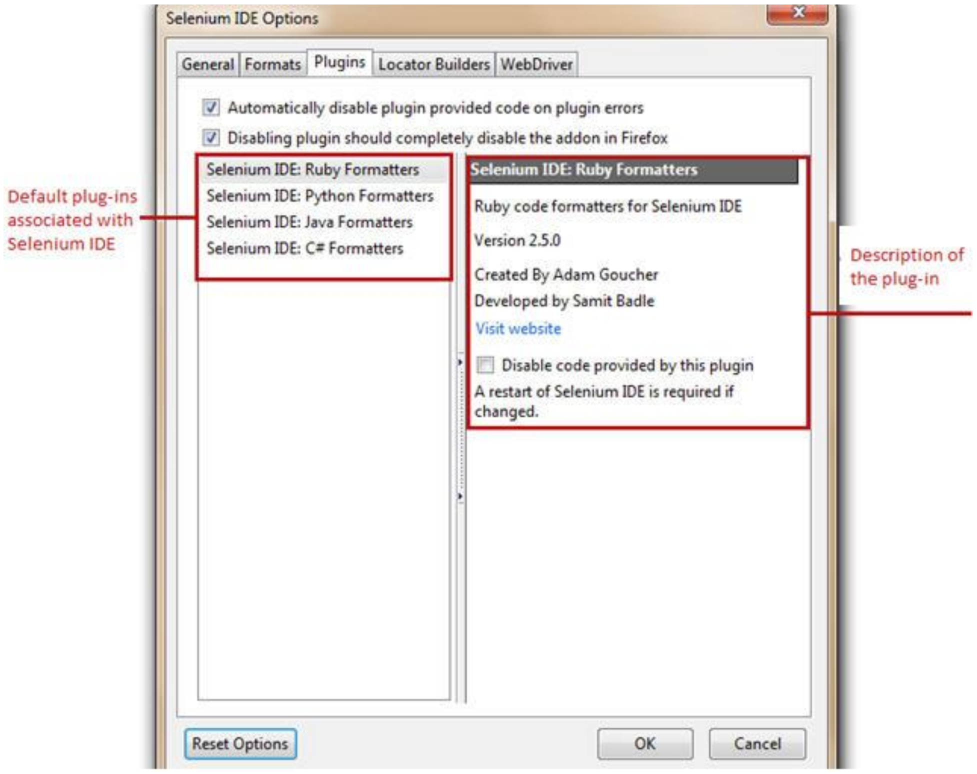
Selenium IDE Plugins

Plug-ins tab displays the supported Firefox plug-ins installed on our instance of Selenium IDE. There are a number of plug-ins available to cater different needs, thus we can install these add-ons like we do other plug-ins. One of the recently introduced plug-in is “File Logging”. In the end of this tutorial, we will witness how to install and use this plug-in.

With the standard distribution, Selenium IDE comes with a cluster of following plug-ins:

- Selenium IDE: Ruby Formatters
- Selenium IDE: Python Formatters
- Selenium IDE: Java Formatters
- Selenium IDE: C# Formatters

These formatters are responsible to convert the HTML test cases into the desired programming formats.



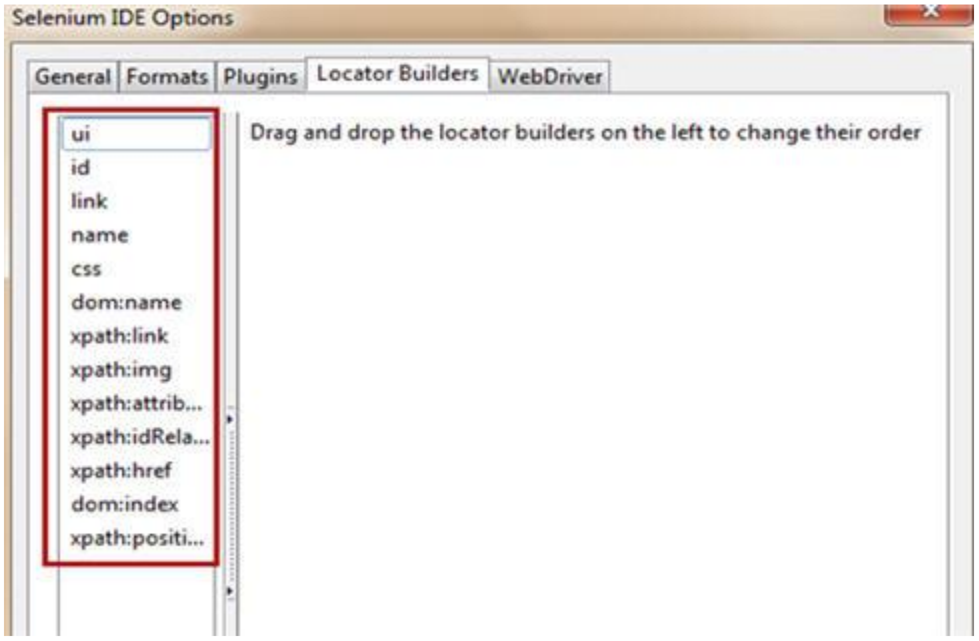
☑ Automatically disable plugin provided code on plugin errors
☑ Disabling plugin should completely disable the addon in Firefox

Selenium IDE: Ruby Formatters
Selenium IDE: Python Formatters
Selenium IDE: Java Formatters
Selenium IDE: C# Formatters

Selenium IDE: Ruby Formatters
Ruby code formatters for Selenium IDE
Version 2.5.0
Created By Adam Goucher
Developed by Samit Badle
Visit website
☐ Disable code provided by this plugin
A restart of Selenium IDE is required if changed.

Locator Builders

Locator builders allow us to prioritize the order of locator types that are generated while recording the user actions. Locators are the set of standards by which we uniquely identify a web element on a web page.

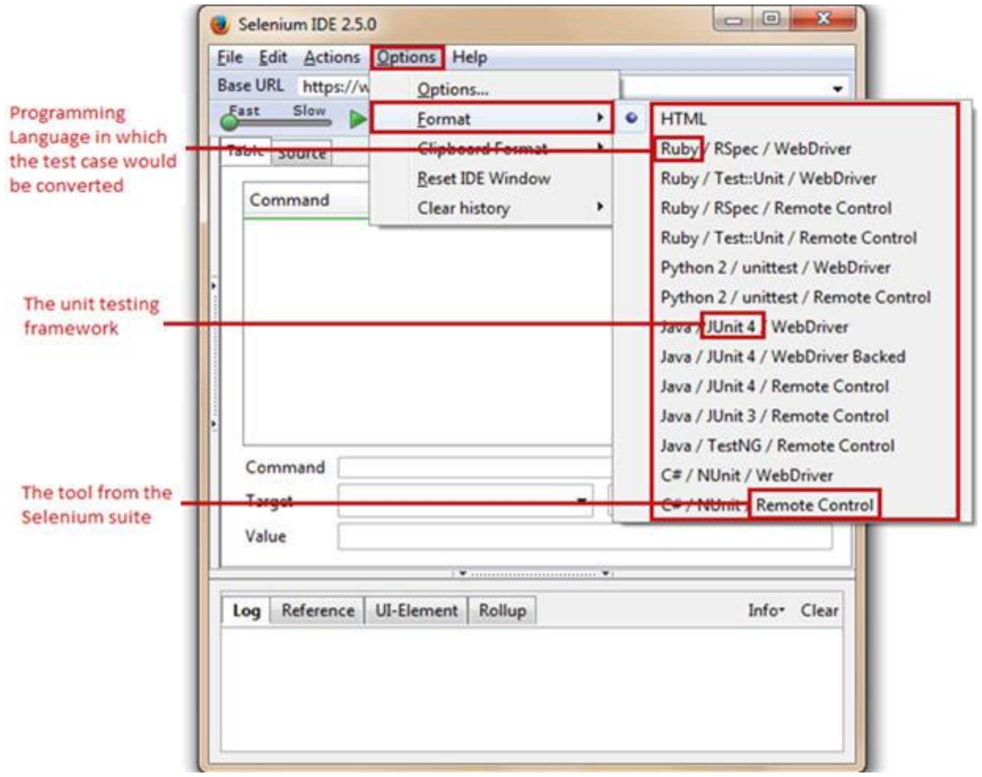


ui
id
link
name
css
dom:name
xpath:link
xpath:img
xpath:attrib...
xpath:idRela...
xpath:href
dom:index
xpath:positi...

Drag and drop the locator builders on the left to change their order

Formats

Formats option allows user to convert the Selenium IDE test case (selenese commands) into desired format.

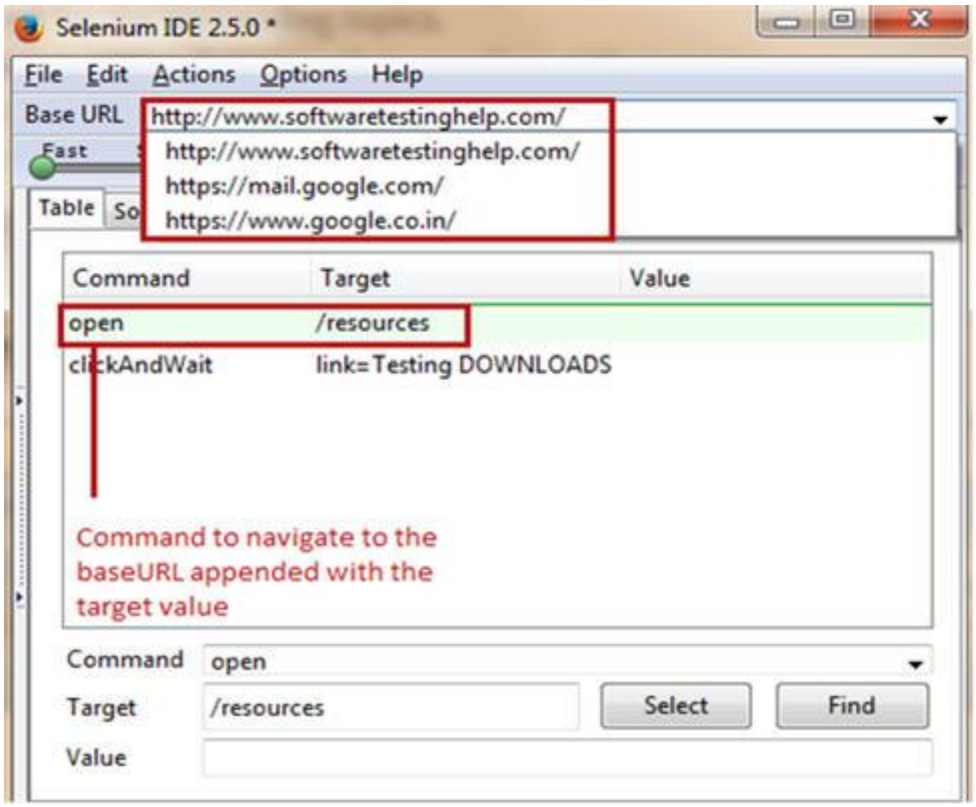

Selenium IDE 2.5.0
File
Edit
Actions
Options
Help
Base URL
https://w
Fast
Slow
Table
Source
Command
Options...
Format
Clipboard Format
Reset IDE Window
Clear history
HTML
Ruby / RSpec / WebDriver
Ruby / Test::Unit / WebDriver
Ruby / RSpec / Remote Control
Ruby / Test::Unit / Remote Control
Python 2 / unittest / WebDriver
Python 2 / unittest / Remote Control
Java / JUnit 4 / WebDriver
Java / JUnit 4 / WebDriver Backed
Java / JUnit 4 / Remote Control
Java / JUnit 3 / Remote Control
Java / TestNG / Remote Control
C# / NUnit / WebDriver
C# / NUnit / Remote Control
Programming Language in which the test case would be converted
The unit testing framework
The tool from the Selenium suite
Command
Target
Value
Log
Reference
UI-Element
Rollup
Info
Clear

E) Help Menu

As Selenium has a wide community and user base, thus various documentations, release notes, guides etc. are handily available. Thus, the help menu lists down official documentation and release notes to help the user.

#2. Base URL Bar

Base URL bar is principally same as that of an address bar. It remembers the previously visited websites so that the navigation becomes easy later on.

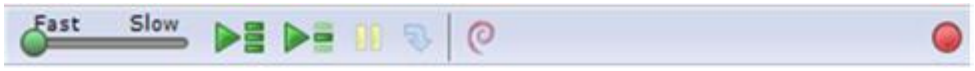

Selenium IDE 2.5.0 *
File Edit Actions Options Help
Base URL
http://www.softwaretestinghelp.com/
http://www.softwaretestinghelp.com/
https://mail.google.com/
https://www.google.co.in/
Fast
Table
Command
Target
Value
open
/resources
clickAndWait
link=Testing DOWNLOADS
Command to navigate to the baseURL appended with the target value
Command
open
Target
/resources
Select
Find
Value

Now, whenever the user uses "open" command of Selenium IDE without a target value, the base URL would be launched on to the browser.

Accessing relative paths

To access relative paths, user simply needs to enter a target value like "/download" along with the "open" command. Thus, the base URL appended with "/downloads" (http://docs.seleniumhq.org/resources) would be launched on to the browser. The same is evident in the above depiction.

#3. Toolbar



Toolbar provides us varied options pertinent to the recording and execution of the test case.

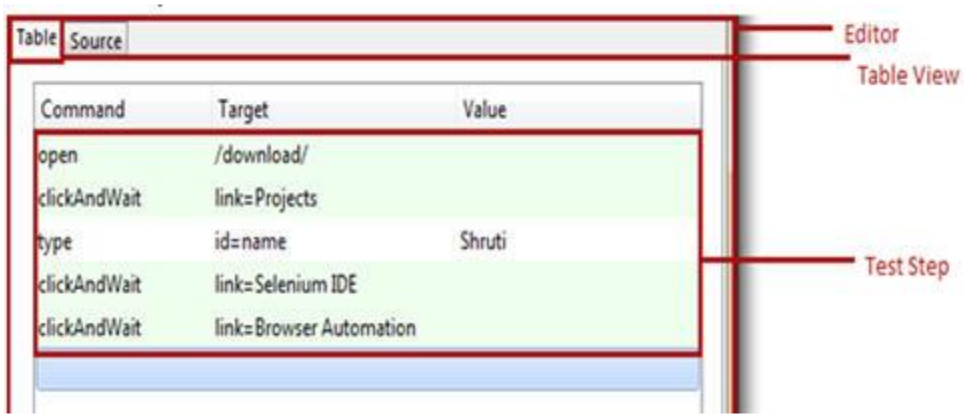
- **Playback Speed** – This option allows user to control the test case execution speed from fast to slow.
- **Play test suite** – This option allows user to execute all the test cases belonging to the current test suite sequentially.
- **Play test case** – This option allows user to execute the currently selected test case.
- **Pause** – This option allows user to pause the current execution.
- **Step** – This option allows user to step into the test step.

- **Rollup**– This option allows user to combine multiple test steps to act like a single command.
- Record – This option allows user to start/stop the recording of user actions. The hollow red ball indicates the start of the recording session whereas the solid red ball indicates the end of the recording session. By default, the Selenium IDE opens in the recording mode.

#4. Editor

Editor is a section where IDE records a test case. Each and every user action is recorded in the editor in the same order in which they are performed.

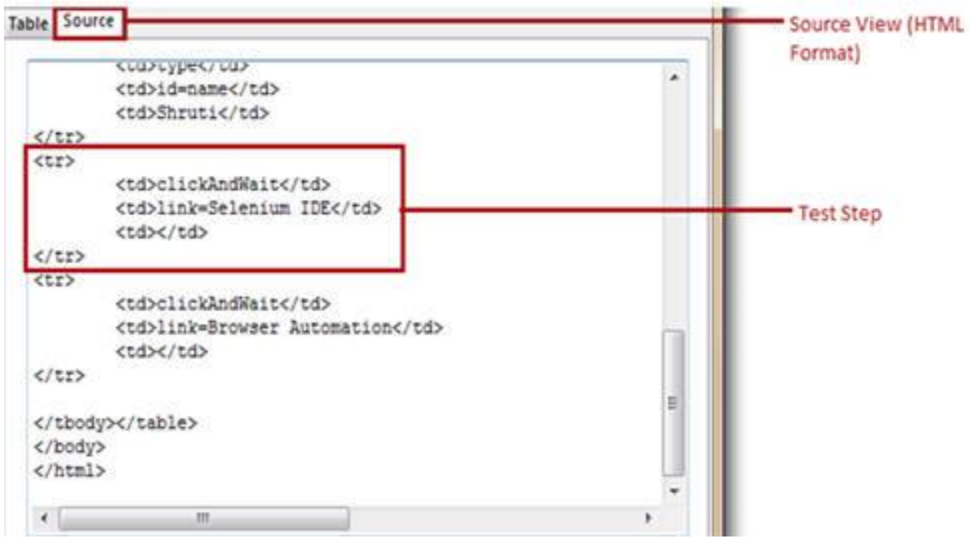
The editor in IDE has two views, namely:
1) Table View



Command	Target	Value
open	/download/	
clickAndWait	link=Projects	
type	id=name	Shruti
clickAndWait	link=Selenium IDE	
clickAndWait	link=Browser Automation	

It is the default view provided by Selenium IDE. The test case is represented in the tabular format. Each user action in the table view is a consolidation of “Command”, “Target” and “Value” where command, target and value refers to user action, web element with the unique identification and test data correspondingly. Besides recording it also allows user to insert, create and edit new Selenese commands with the help of the editor form present in the bottom.

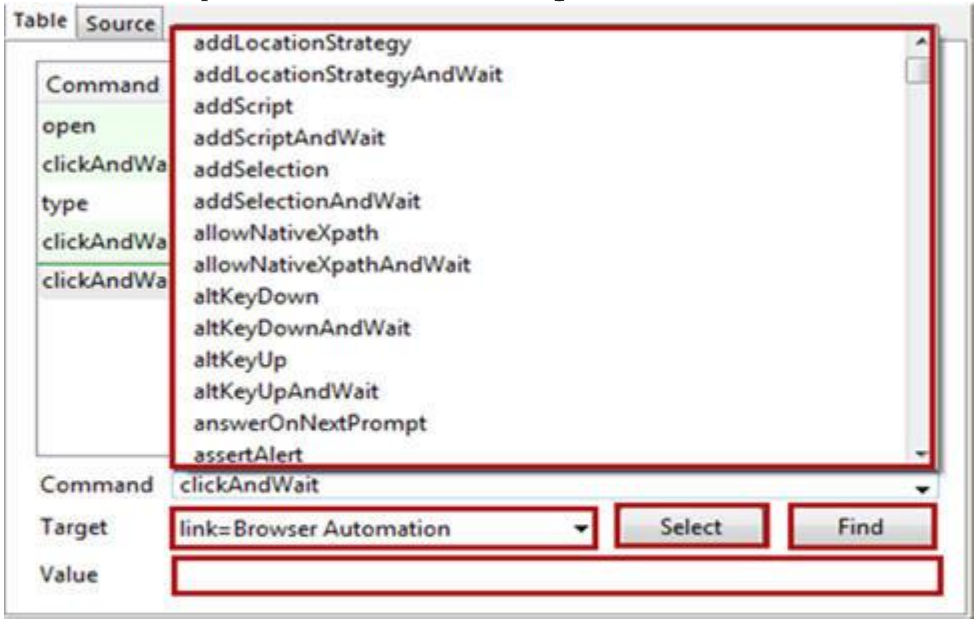
2) Source View



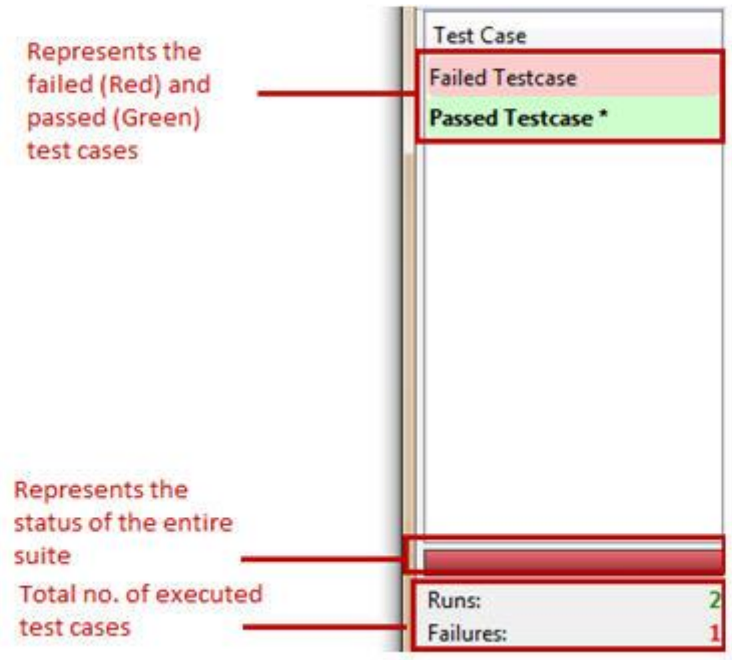
```
        <td>type</td>
        <td>id=name</td>
        <td>Shruti</td>
</tr>
<tr>
        <td>clickAndWait</td>
        <td>link=Selenium IDE</td>
        <td></td>
</tr>
<tr>
        <td>clickAndWait</td>
        <td>link=Browser Automation</td>
        <td></td>
</tr>

</tbody></table>
</body>
</html>
```

The test case is represented in the HTML format. Each test step is considered be a row <tr> which is a combination of command, target and value in the separate columns <td>. Like any HTML document, more rows and columns can be added to correspond to each Selenese command.

Editor Form lets the user to type any command and the suggestions for the related command would be populated automatically. Select button lets the user to select any web element and its locator would be fetched automatically into the target field. Find button lets the user find the web element on the web page against a defined target. Value is the test input data entered into the targets with which we want to test the scenario.



#5. Test case pane


Represents the failed (Red) and passed (Green) test cases
Test Case
Failed Testcase
Passed Testcase *
Represents the status of the entire suite
Total no. of executed test cases
Runs: 2
Failures: 1

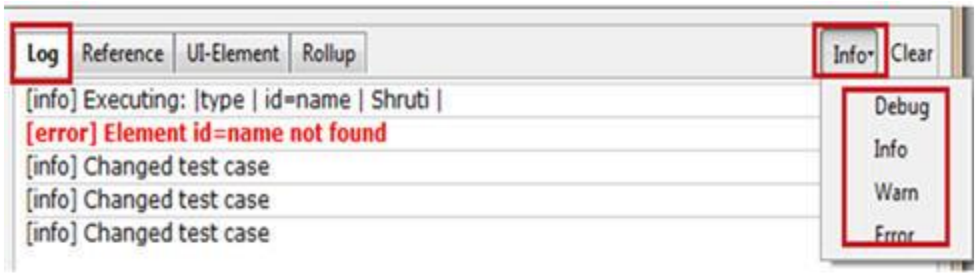
At the instance we open Selenium IDE interface, we see a left container titled "Test case" containing an untitled test case. Thus, this left container is entitled as Test case pane.

Test case pane contains all the test cases that are recorded by IDE. The tool has a capability of opening more than one test case at the same time under test case pane and the user can easily shuffle between the test cases. The test steps of these test cases are organized in the editor section.

Selenium IDE has a color coding ingredient for reporting purpose. After the execution, the test case in marked either in "red" or "green" color.

- Red color symbolizes the unsuccessful run i.e. failure of the test case.
- Green color symbolizes the successful run of the test case
- It also layouts the summary of the total number of test cases executed with the number of failed test cases.
- If we execute a test suite, all the associated test cases would be listed in the test case pane. Upon execution, the above color codes would be rendered accordingly.

#6. Log Pane



```
[info] Executing: |type | id=name | Shruti |
[error] Element id=name not found
[info] Changed test case
[info] Changed test case
[info] Changed test case
```

Log pane gives the insight about current execution in the form of messages along with the log level in the real time. Thus, log messages enable a user to debug the issues in case of test case execution failures.

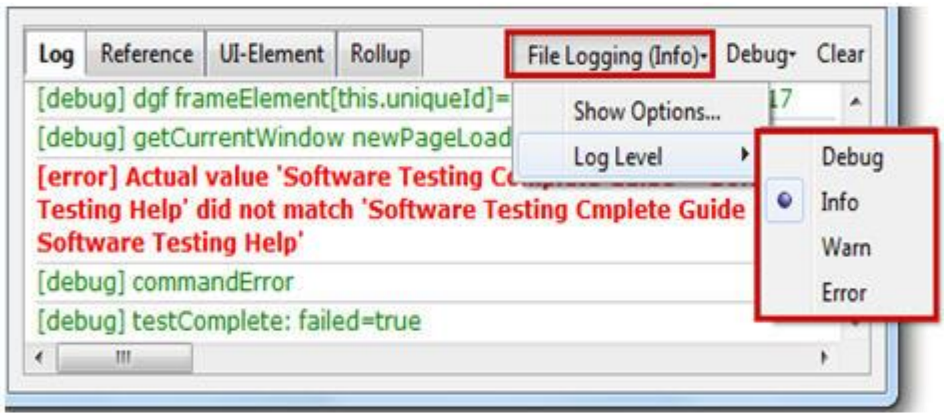
The printing methods / log levels used for generating logs are:

- Error – Error message gives information about the test step failure. It may be generated in the cases when element is not found, page is not loaded, verification/assertion fails etc.
- Warn – Warning message gives information about unexpected conditions.
- Info – Info message gives information about current test step execution.
- Debug – Debug messages gives information about the technicalities in the backdrop about the current test step.

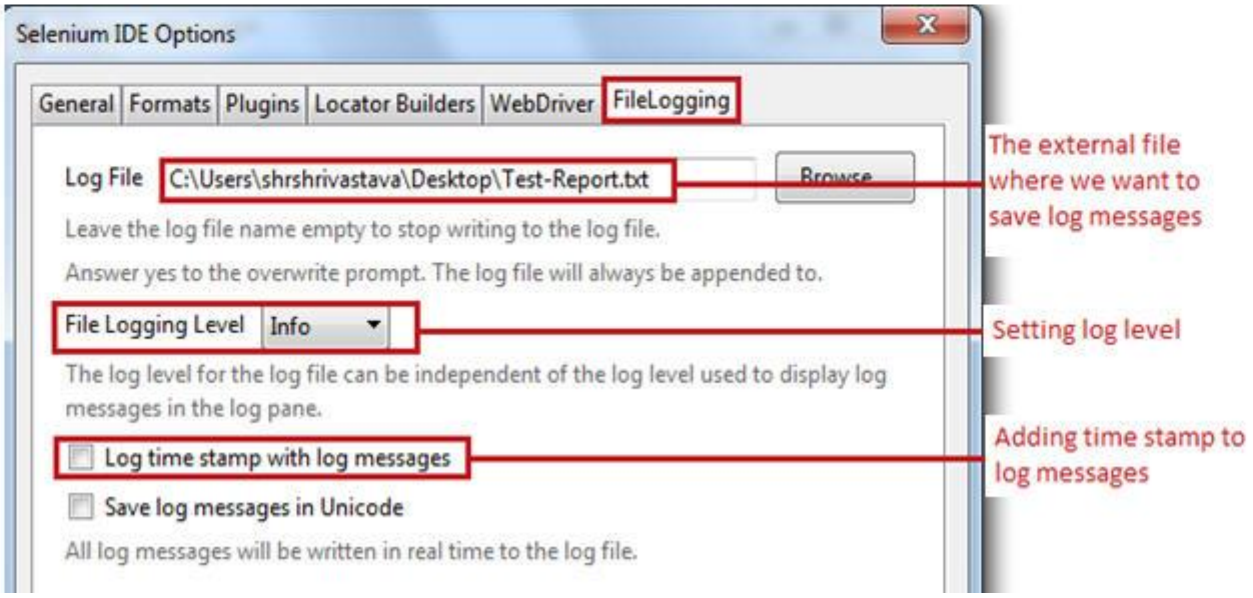
Logs can be filtered with the help of a drop down located at the top-right corner of the footer beside the clear button. Clear button erases all the log messages generated in the current or previous run.

Generating Logs in an external medium

Recently introduced “File Logging” plug-in enables the user to save log messages into an external file. File Logging can be plugged in to IDE like any other plug-in. Upon installation, it can be found as a tab named “File Logging” in the footer beside the Clear button.



```
[debug] dgf frameElement[this.uniqueId]=                17
[debug] getCurrentWindow newPageLoad
[error] Actual value 'Software Testing Co
Testing Help' did not match 'Software Testing Cmplete Guide
Software Testing Help'
[debug] commandError
[debug] testComplete: failed=true
```


Selenium IDE Options

Log File C:\Users\shrshrivastava\Desktop\Test-Report.txt

Leave the log file name empty to stop writing to the log file.

Answer yes to the overwrite prompt. The log file will always be appended to.

File Logging Level Info

The log level for the log file can be independent of the log level used to display log messages in the log pane.

☐ Log time stamp with log messages

☐ Save log messages in Unicode

All log messages will be written in real time to the log file.

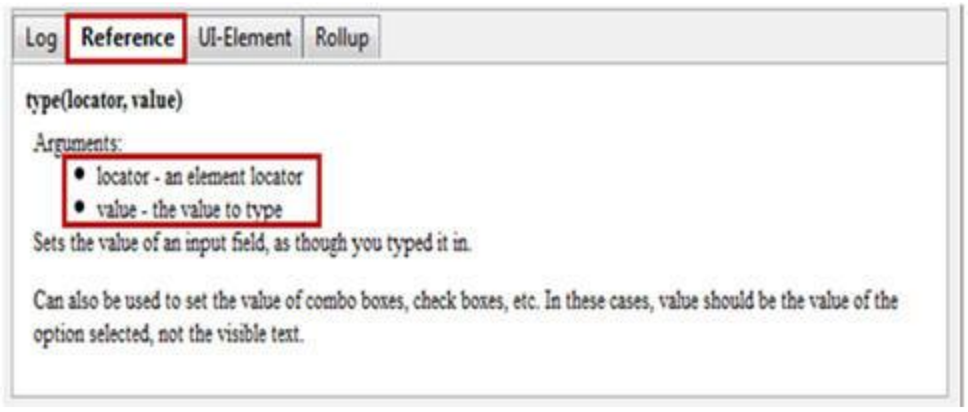
The external file where we want to save log messages

Setting log level

Adding time stamp to log messages

Reference Pane

Reference Pane gives the brief description about the currently selected Selenese command along with its argument details.



type(locator, value)

Arguments:
- locator - an element locator
- value - the value to type

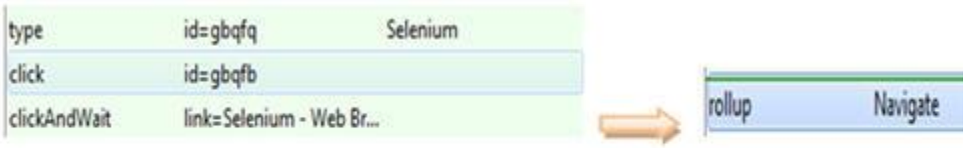
Sets the value of an input field, as though you typed it in.

Can also be used to set the value of combo boxes, check boxes, etc. In these cases, value should be the value of the option selected, not the visible text.

UI-Element Pane

UI – Element Pane allows Selenium user to use JavaScript Object Notation acronym as JSON to access the page elements. More on this can be found in UI-Element Documentation under Help Menu.

Rollup Pane



type	id=gbqfq	Selenium
click	id=gbqfb	
clickAndWait	link=Selenium - Web Br...	

rollup	Navigate

Rollup Pane allows the user to roll up or combine multiple test steps to constitute a single command termed as “rollup”. The rollup in turn can be called multiple times across the test case.

Conclusion

Through this tutorial, our objective was to make you familiar and accustomed with the basic terminologies and nomenclatures of Selenium IDE. We also presented a detailed study on all the features of Selenium IDE.

Here are the cruxes of this tutorial:

- Selenium IDE is an automated testing tool which supports record and play back.
- User is not required to have any prior programming knowledge except for the basic understanding of HTML, JavaScript and DOM.
- The menu bar allows user to create, save, edit and convert the recorded Selenium IDE test scripts. It also allows user to set formats and plug-ins.
- Toolbar allows user to set the test execution speed, to pause and resume test case, to roll up commands etc.
- Roll ups combines more than one test step and thus the rolled up commands acts and executes as a single command.
- Editor allows user to record or create test scripts. Editor has two views “table” and “source”.
- In table view, each test step is comprised of a command, target and a value.
- Source view displays the test case in the HTML format.
- Test case pane shows a comprehensive list of failed and passed test cases with the relevant color-coding.
- Log Pane displays the test execution heath in the form of message.
- Log messages can be saved in a file using “File Logging” plug-in.
- Reference pane shows the description of every selected command.
- UI-Element and Rollup are generally used while creating advance Selenium IDE scripts.

Next Tutorial #3: Now that we are acquainted and comfortable with Selenium IDE and its features, in the next tutorial we would practice these features by creating our own test script using Selenium IDE.

A remark for the readers: While our next tutorial of the Selenium series is in the processing mode, install the tool and the required utilities to get started. Experience the features by playing around with the tool till we meet next with the next tutorial ***on “My first Selenium IDE script”.***

Stay tuned till then and share your views, comments and knowledge to help us groom. Also let us know if you find anything that we missed out so that we can include them in the subsequent tutorials.

Finally, if you like this tutorial please consider sharing it with friends and on social media sites.

My First Selenium IDE Script – Selenium Tutorial #3

***This tutorial is by far one of the most important tutorials to get a hold on Selenium IDE*.**
*This is the 3rd tutorial in our multi-part **Selenium Tutorials series**. We started this Selenium online Training series from this tutorial where you can find list of all tutorials covered.*
In the introductory tutorials, we got a brief idea about Selenium IDE and its features.
Going ahead, we would be exercising and implementing these features in real time by creating our own very first Selenium IDE script. We would also peek into the details of recording fundamentals and available types of commands in Selenium IDE. Apart from that we would also have a glance at the modifications which can be incorporated into our Selenium scripts.

Before jumping on to the creation of Selenium IDE script, let us take a moment to introduce elementary information about the application under test (AUT).

As a specimen, we would be using "Gmail" – an email service designed by Google. I believe because of its unbounded popularity, it needs no more introductions. The URL we would be using is "https://accounts.google.com". I have also created dummy credentials to represent test data.

Creating First Selenium IDE Script

So let us now create our first script using Selenium IDE.

The entire script creation process can be classified into 3 chunks:
Process #1: Recording – Selenium IDE aids the user to record user interactions with the browser and thus the recorded actions as a whole are termed as Selenium IDE script.
Process #2: Playing back – In this section, we execute the recorded script so as to verify and monitor its stability and success rate.
Process #3: Saving – Once we have recorded a stable script, we may want to save it for future runs and regressions.
Let us now see their implementation.

Process #1: Recording a test script

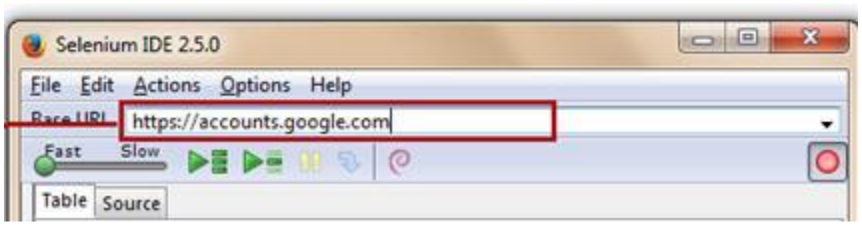
Scenario

- Open "https://accounts.google.com".
- Assert Title of the application
- Enter a valid username and password and submit the details to login.

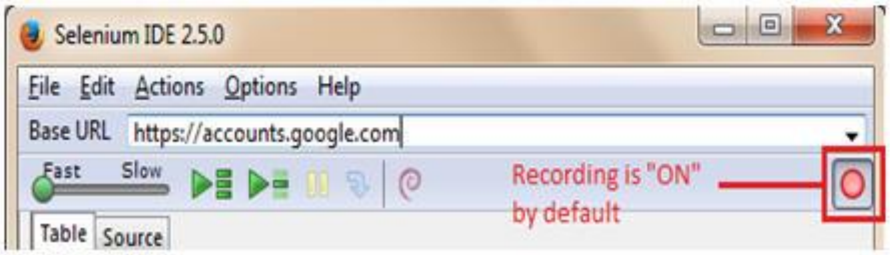
- Verify that the user is re-directed to the Home page.

Step 1 – Launch the Firefox and open Selenium IDE from the menu bar.

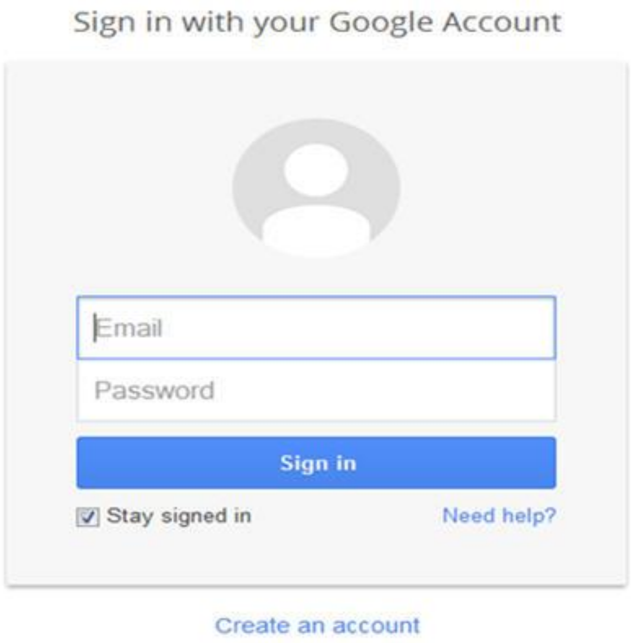
Step 2 – Enter the address of application under test (“https://accounts.google.com”) inside the Base URL textbox.



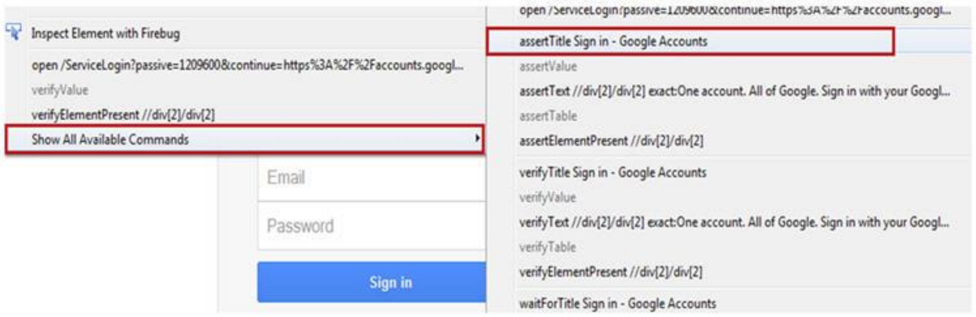
Step 3 – By default, the Record button is in ON state. Remember to tune it ON if it is in OFF state so as to enable the recording mode.


Selenium IDE 2.5.0
File Edit Actions Options Help
Base URL
https://accounts.google.com
Fast Slow
Recording is "ON" by default
Table Source

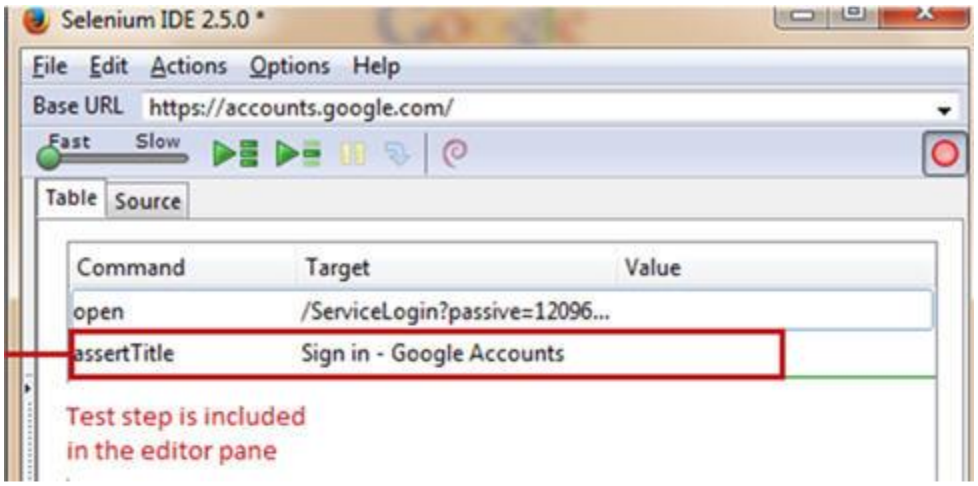
Step 4 – Open the application under test (https://accounts.google.com) in the Firefox.



Step 5 – Verify if the application title is correct. To do so, right click anywhere on the page except the hyperlinks or images. The right click opens the Selenium IDE context menu listing few of the commands. To get an entire list, select “Show Available Commands” option. This will open another menu containing rest of the available and applicable commands. Select “assertTitle Sign in – Google Accounts” option to verify the page title.


Inspect Element with Firebug
open /ServiceLogin?passive=1209600&continue=https%3A%2F%2Faccounts.googl...
verifyValue
verifyElementPresent //div[2]/div[2]
Show All Available Commands
Email
Password
Sign in
open /ServiceLogin?passive=1209600&continue=https%3A%2F%2Faccounts.googl...
assertTitle Sign in - Google Accounts
assertValue
assertText //div[2]/div[2] exact:One account. All of Google. Sign in with your Googl...
assertTable
assertElementPresent //div[2]/div[2]
verifyTitle Sign in - Google Accounts
verifyValue
verifyText //div[2]/div[2] exact:One account. All of Google. Sign in with your Googl...
verifyTable
verifyElementPresent //div[2]/div[2]
waitForTitle Sign in - Google Accounts

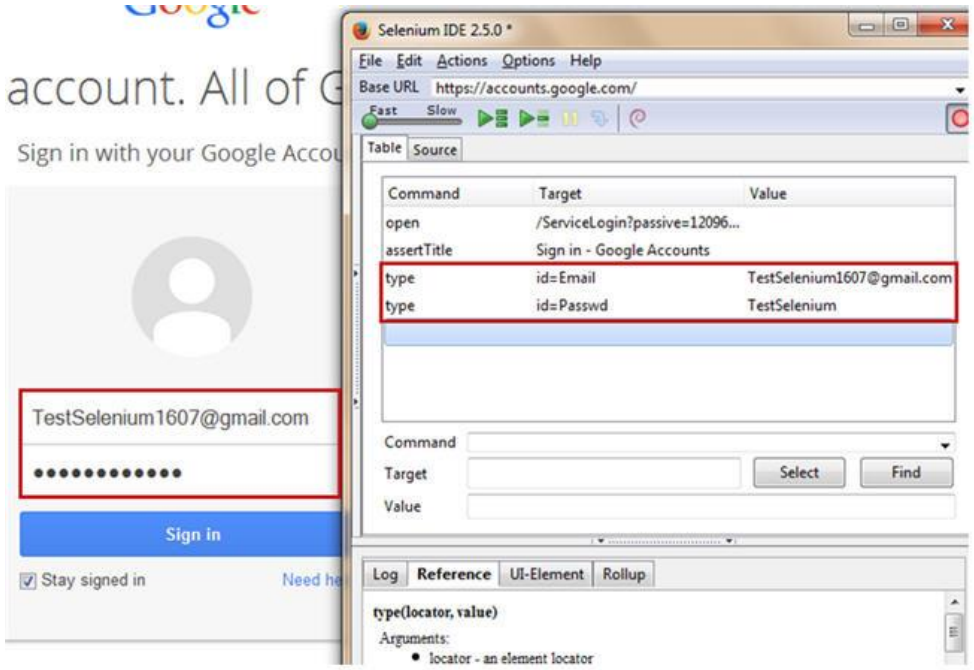
As soon as we click on "assertTitle Sign in – Google Accounts" option, a test step would be included /appended in the Selenium IDE editor.



Command	Target	Value
open	/ServiceLogin?passive=12096...	
assertTitle	Sign in - Google Accounts	

Step 6 – Enter a valid username in the "Email" Textbox of Gmail.
Step 7 – Enter a valid password in the "Password" Textbox of Gmail.
The simulation of the same user actions can be seen in the Selenium IDE test editor.

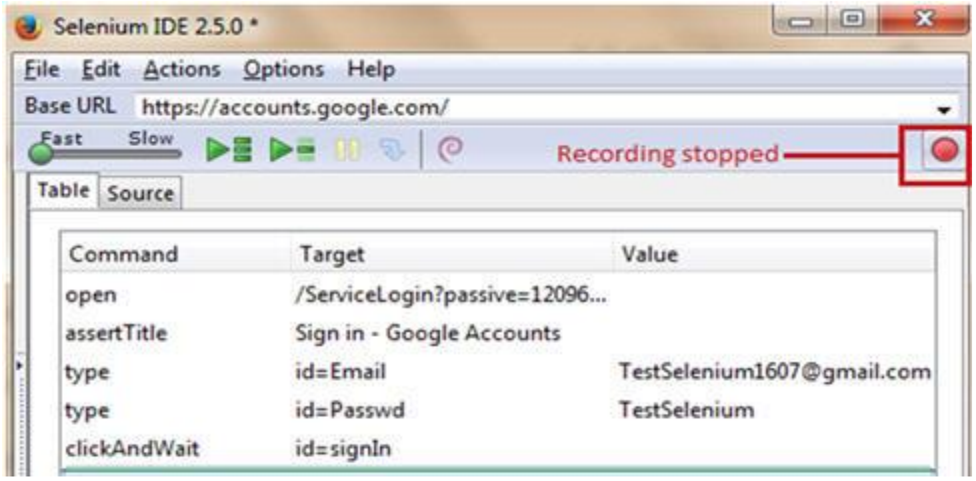
Notice that for the ease of understanding, I have already created test credentials. I would strictly advise the readers to create their own credentials instead of using these.


account. All of G
Sign in with your Google Accou
TestSelenium1607@gmail.com
••••••••••••
Sign in
Stay signed in
Need he
Selenium IDE 2.5.0 *
File Edit Actions Options Help
Base URL https://accounts.google.com/
Fast Slow
Table Source
Command
Target
Value
Select
Find
Log Reference UI-Element Rollup
type(locator, value)
Arguments:
locator - an element locator

Command	Target	Value
open	/ServiceLogin?passive=12096...	
assertTitle	Sign in - Google Accounts	
type	id=Email	TestSelenium1607@gmail.com
type	id=Passwd	TestSelenium

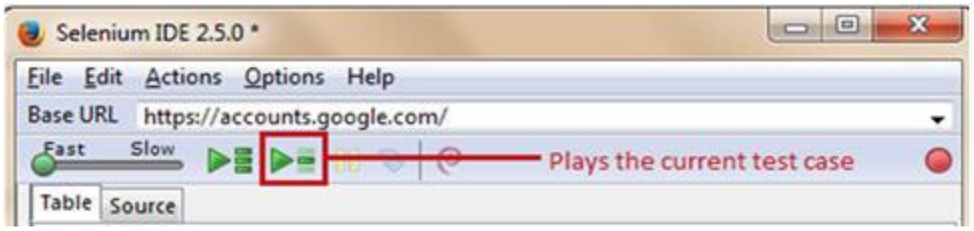
Step 8 – Click on the "Sign in" button to complete the login process.
User should be re-directed to the home page provided the credentials entered are correct.

Step 9 – At the end, we would end the recording session by tuning the record button into OFF state. Below is the recorded script.

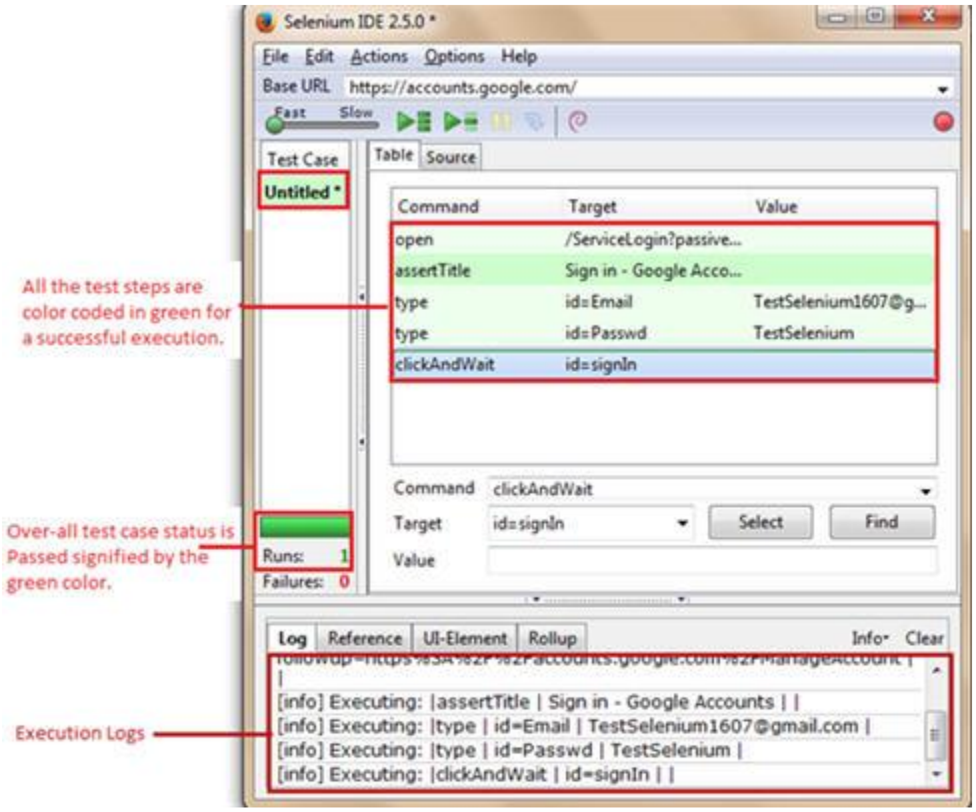

Selenium IDE 2.5.0 *
File Edit Actions Options Help
Base URL https://accounts.google.com/
Fast Slow
Recording stopped
Table Source

Command	Target	Value
open	/ServiceLogin?passive=12096...	
assertTitle	Sign in - Google Accounts	
type	id=Email	TestSelenium1607@gmail.com
type	id=Passwd	TestSelenium
clickAndWait	id=signIn	

Process #2: Playing back / executing a test script
Now that we have created our first Selenium IDE script, we would want to execute it to see if the script is stable enough. Click on the playback button to execute the script.


Selenium IDE 2.5.0 *
File Edit Actions Options Help
Base URL https://accounts.google.com/
Fast Slow
Plays the current test case
Table Source

Post execution, all the test steps would be color coded in green for the successful run. The same would be evitable from the test case pane.


Selenium IDE 2.5.0 *
File Edit Actions Options Help
Base URL https://accounts.google.com/
Fast Slow
Test Case
Untitled *
Table Source
All the test steps are color coded in green for a successful execution.
Over-all test case status is Passed signified by the green color.
Execution Logs
Runs: 1
Failures: 0
Command clickAndWait
Target id=signIn
Value
Select Find
Log Reference UI-Element Rollup Info Clear

Command	Target	Value
open	/ServiceLogin?passive...	
assertTitle	Sign in - Google Acco...	
type	id=Email	TestSelenium1607@g...
type	id=Passwd	TestSelenium
clickAndWait	id=signIn	

```
[info] Executing: |assertTitle | Sign in - Google Accounts | |
[info] Executing: |type | id=Email | TestSelenium1607@gmail.com |
[info] Executing: |type | id=Passwd | TestSelenium |
[info] Executing: |clickAndWait | id=signIn | |
```

For unsuccessful execution or test case failure, the failed test step would be highlighted in red. And the test case pane would mark the test case execution as failure.

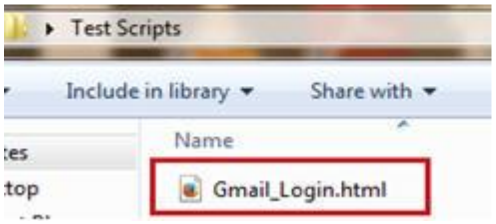
Process #3: Saving a test script

Once, we have played back the script, now it's time to save the created test script.

Step 1 – To save the test script, Click on the File menu and select "Save Test Case" option.

Step 2 – The system will prompt us to browse or enter the desired location to save our test case and to provide the test script name. Furnish the test name as "Gmail_Login" and click on the "Save" button.

The test script can be found at the location provided in the above step. Notice that the test script is saved in HTML format.

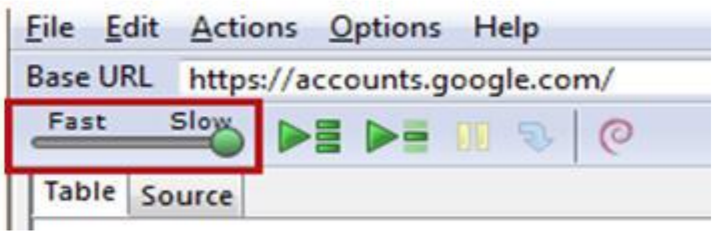


Using Common features of Selenium IDE

Setting Execution speed

While testing web applications, we come across several scenarios where an action performed may trigger a page load. Thus we must be cognizant enough while dealing such scenarios.

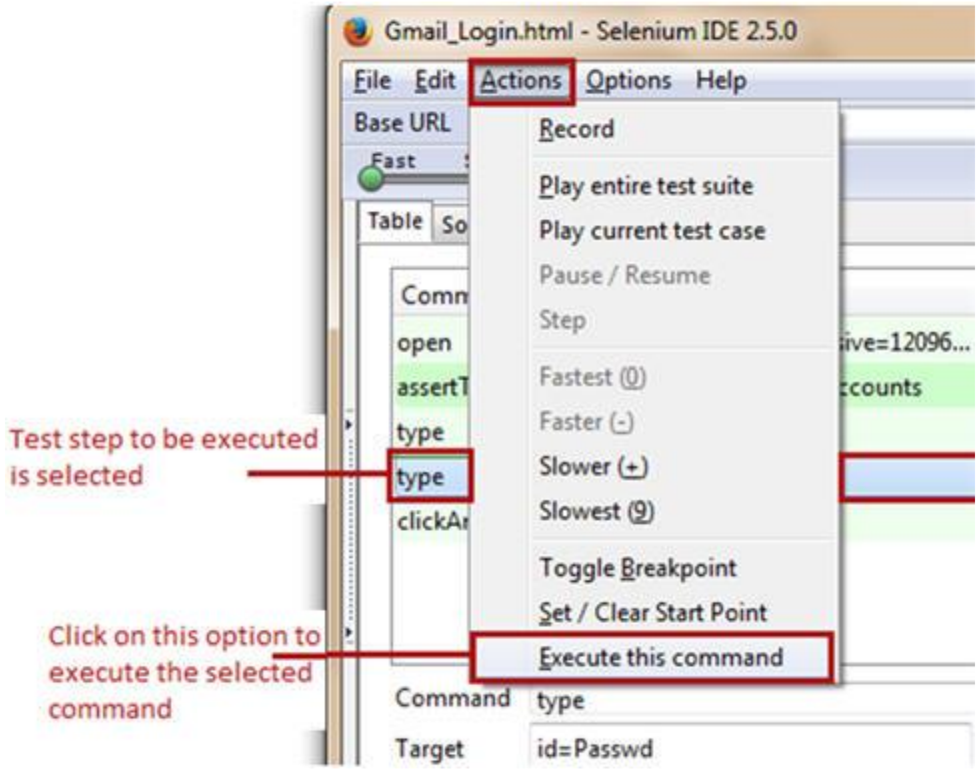
So to avoid failures while playing back these test scenarios, we can set the execution speed to be minimal. Refer the following figure for the same.


File Edit Actions Options Help
Base URL https://accounts.google.com/
Fast Slow
Table Source

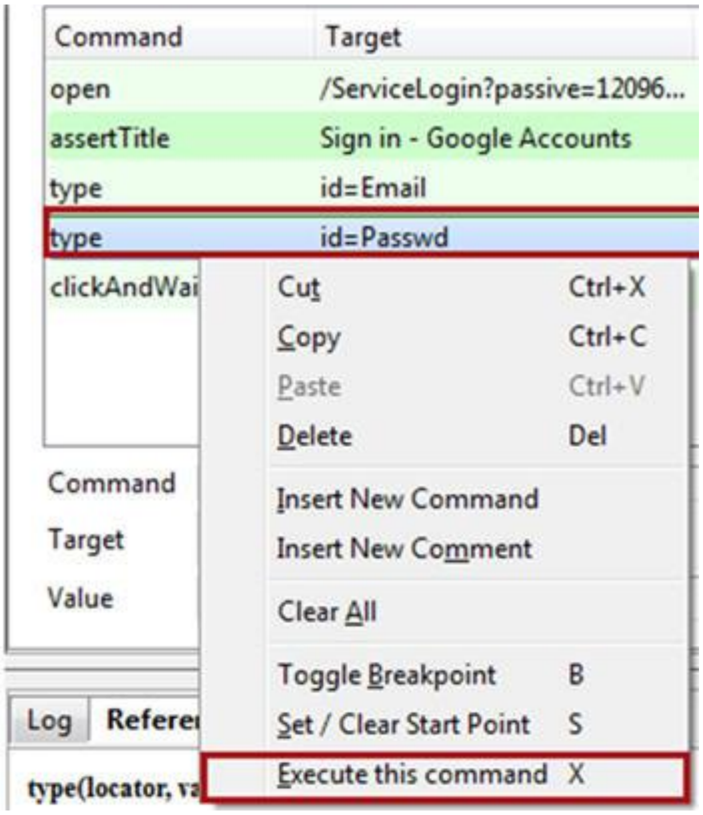
Using "Execute this command" option
Selenium IDE allows the user to execute a single test step within the entire test script without executing the entire test script. "Execute this command" is the option which makes this obtainable.

"Execute this command" option can be used at times when we want to debug/see the behavior of a particular test step.

"Execute this command" option can be used in the following four ways:
#1. Using Actions tab from the Menu bar



#2. Using short cut key ("X")
#3. Right click the test step and select "Execute this command"


Command	Target
open	/ServiceLogin?passive=12096...
assertTitle	Sign in - Google Accounts
type	id=Email
type	id=Passwd
clickAndWai	

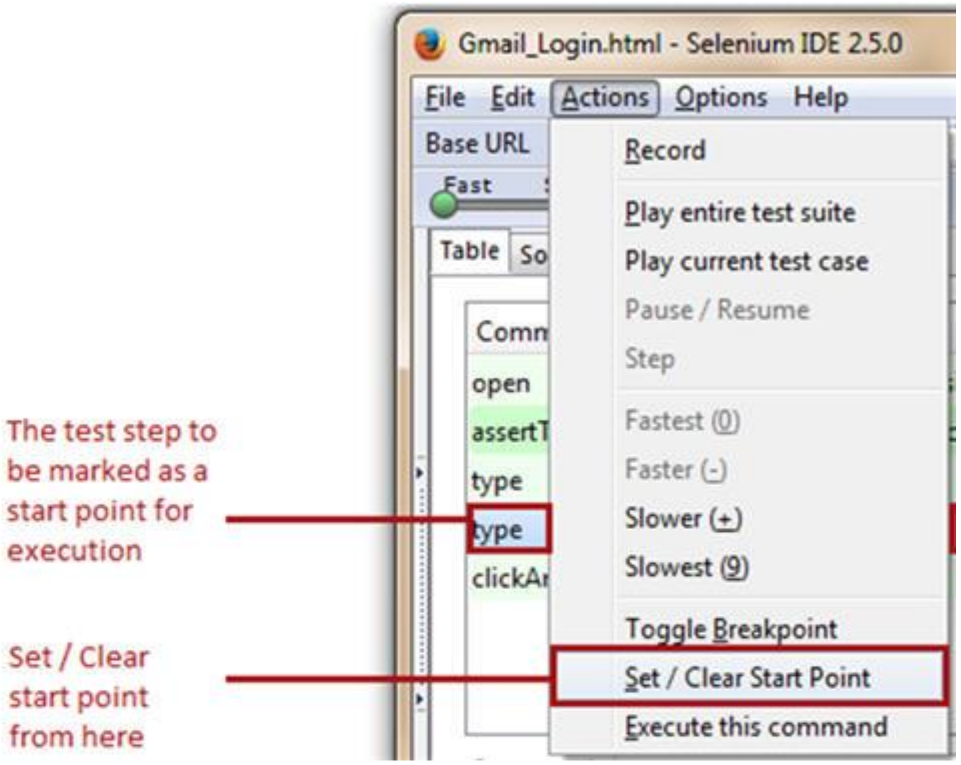
#4. Double click the test step
In all the above cases, user is expected to select the test step which he / she want to execute.

Steps to be followed:
Step 1 – Launch the web browser and open the target URL ("https://accounts.google.com"), Select the test step that we desire to execute. Remember to open correct web page to mitigate the chances of failure.
Step 2 – Press "X" to execute the selected test step. Alternatively, one can use other ways too.
Step 3 – Notice that the selected test step is executed. The test step would be color coded in green for success or red for failure. At the same time, the test step would be simulated into an action on the web browser.
Note that the user is responsible to bring the script before executing the test step and Firefox in context. There is a probability of failure if the user has not opened the legitimate web page.

Using Start point
Selenium IDE allows the user to specify a start point within a test script. The start point points to the test step from where we wish to start the test script execution.

Start point can be used at times when we do not desire to execute the entire test script starting from the beginning rather we customize the script to execute from a certain step.

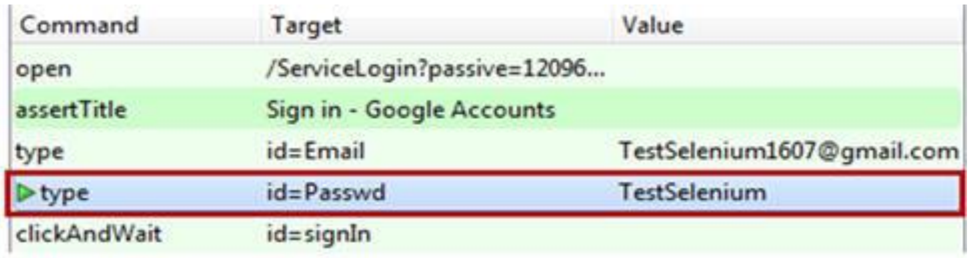
Start point can be set and clear in the following three ways:
#1. Using Actions tab from the Menu bar



#2. Using short cut key ("S")

#3. Right click the test step and select "Set/Clear Start Point". Menu similar to above image will be displayed. In all the above cases, user is expected to select the test step from where he wants to start the execution prior to setting start point.

As soon as the user has marked the test step to indicate the start point, an icon gets affixed to it.



Command	Target	Value
open	/ServiceLogin?passive=12096...	
assertTitle	Sign in - Google Accounts	
type	id=Email	TestSelenium1607@gmail.com
▷type	id=Passwd	TestSelenium
clickAndWait	id=signIn	

Now whenever we execute the test script, it execution would be started from the start point i.e. fourth line (type | id=Passwd | TestSelenium) of the test script.

Notes

- There can be one and only one start point in a single script.
- The start point can be cleared in the same way it was set.
- User is responsible to bring the script after applying start point and Firefox in context. There is a probability of failure if the user has not opened the legitimate web page.

Using Break point

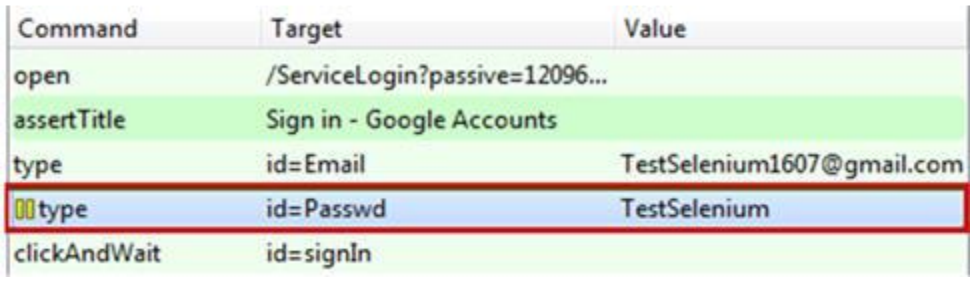
Selenium IDE allows the user to specify break points within a test script. The break points indicate Selenium IDE where to pause the test script.

Break points can be used at times when we desire to break the execution in smaller logical chunks to witness the execution trends.

Break point can be set and clear in the following three ways:

- Using Actions tab from the Menu bar
- Right click the test step and select "Toggle Breakpoint".
- Using short cut key ("B")

As soon as the user has marked the test step to indicate the break point, an icon gets affixed to it.

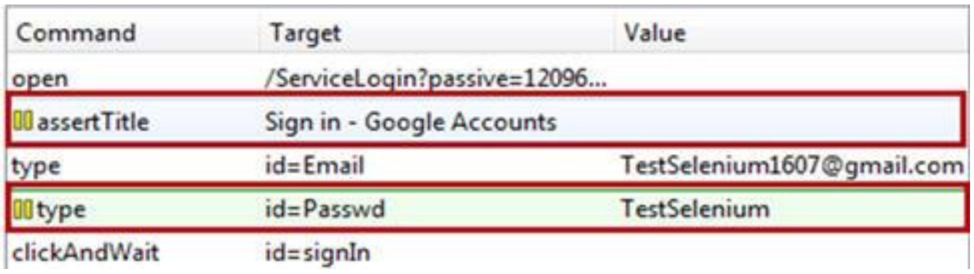


Command	Target	Value
open	/ServiceLogin?passive=12096...	
assertTitle	Sign in - Google Accounts	
type	id=Email	TestSelenium1607@gmail.com
type	id=Passwd	TestSelenium
clickAndWait	id=signIn	

Now whenever we execute the test script, the execution pauses at the break point i.e. fourth line (type | id=Passwd | TestSelenium) of the test script.

Apply multiple breakpoints

Selenium IDE allows user to apply multiple breakpoints in a single test script. Once the first section of the test script is executed, the script pauses as and when the breakpoint is reached. To execute the subsequent test steps, user is required to execute each of the test steps explicitly.



Command	Target	Value
open	/ServiceLogin?passive=12096...	
assertTitle	Sign in - Google Accounts	
type	id=Email	TestSelenium1607@gmail.com
type	id=Passwd	TestSelenium
clickAndWait	id=signIn	

In the above test script, the execution pauses at the line "assertTitle | Sign in – Google Accounts". After explicitly executing this test step, the control moves to the next test step in sequence "type | id=Email | TestSelenium1607@gmail.com". Thus, user needs to explicitly execute this test step. The similar trend is followed for rest of the subsequent steps.

Thus, this feature lets the user to spend more time executing each step and reviewing the outcomes of the previously executed test step.

Notes

- There can be as many break points as you wish in a single script.
- The break point can be cleared in the same way it was set.

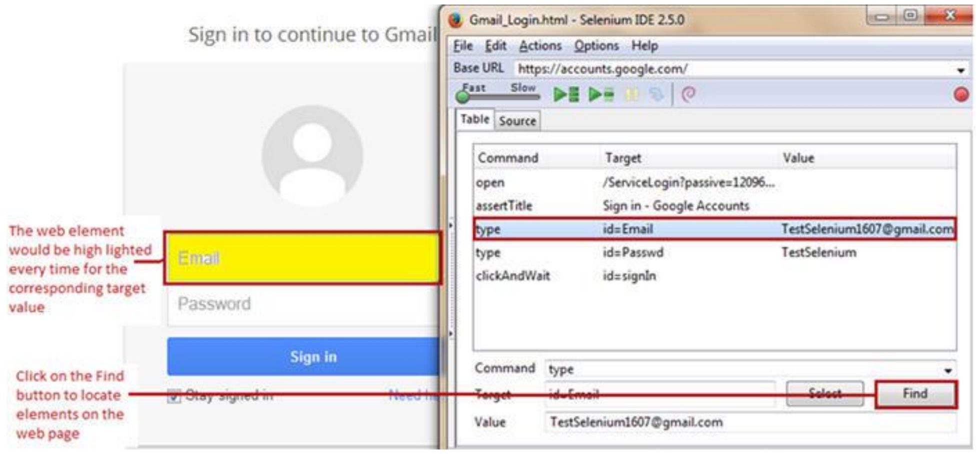
Using Find Button

One of the most crucial aspects of Selenium IDE test scripts is to find and locate web elements within a web page. At times, there are web elements which have analogous properties associated with them, thus making it challenging for a user to identify a particular web element uniquely.

To address this issue, Selenium IDE provides Find button. The Find Button is used to ascertain that locator value provided in the Target test box is indeed correct and identifies the designated web element on the GUI.

Let us consider the above created Selenium IDE test script. Select any command and notice the target text box. Click on the Find button present just beside the Target text box.

Notice that the corresponding web element would be highlighted in yellow with a fluorescent green border around it. If no or wrong web element is highlighted, then the user is required to rectify the issue and would need to impose some other locator value.



Gmail_Login.html - Selenium IDE 2.5.0

Base URL https://accounts.google.com/

Command	Target	Value
open	/ServiceLogin?passive=12096...	
assertTitle	Sign in - Google Accounts	
type	id=Email	TestSelenium1607@gmail.com
type	id=Passwd	TestSelenium
clickAndWait	id=signIn	

Command: type
Target: id=Email
Value: TestSelenium1607@gmail.com

Sign in to continue to Gmail
Email
Password
Sign in
Stay signed in
The web element would be high lighted every time for the corresponding target value
Click on the Find button to locate elements on the web page
File Edit Actions Options Help
Fast Slow
Table Source
Select
Find

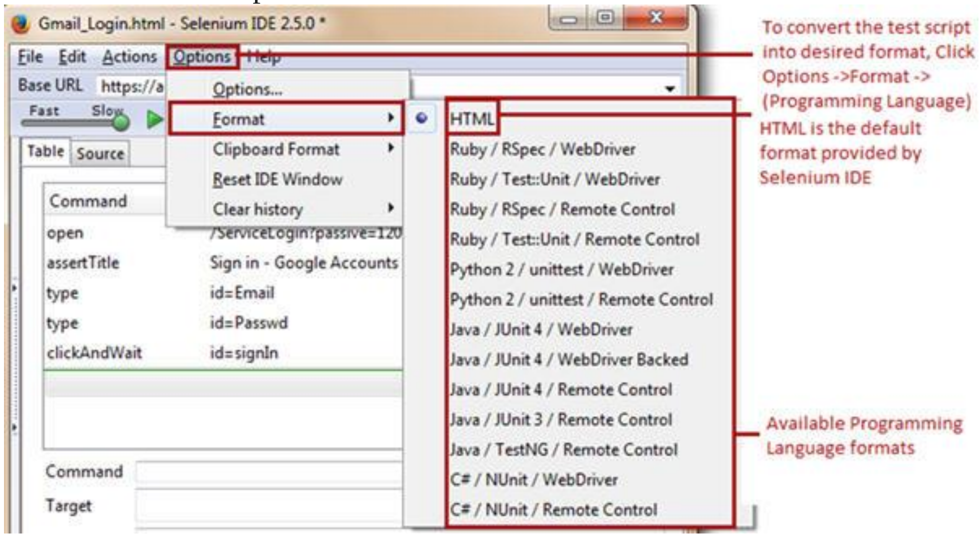
Thus, this procedure makes the user assured about the target value being used and that it corresponds to the correct web element on the GUI.

Using Other Formats

Converting Selenium IDE test scripts to Other Programming Languages

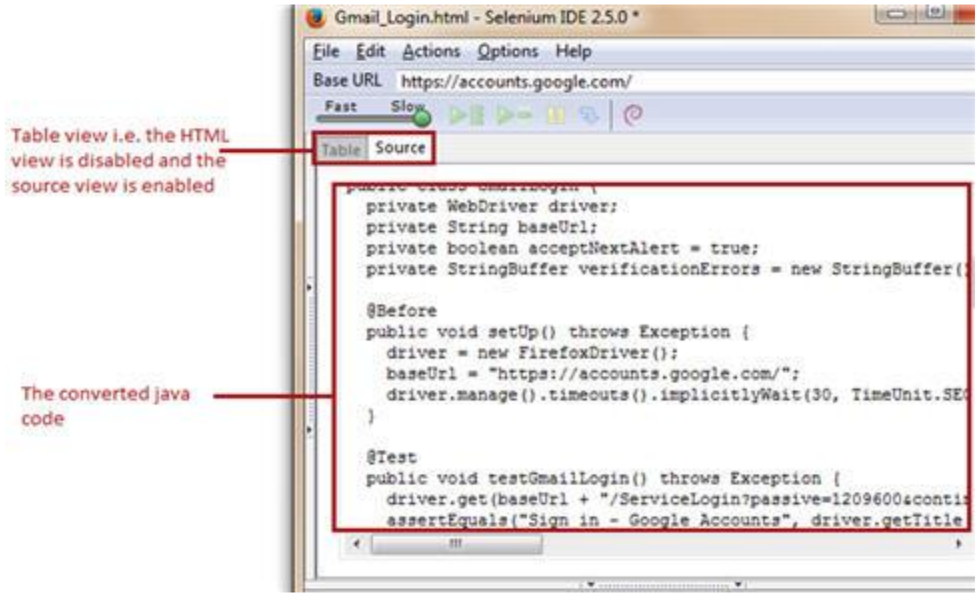
Selenium IDE supports conversion test scripts into set of programming languages from a default type (HTML). The converted test scripts cannot be played back using Selenium IDE until and unless it is reverted back to HTML. Thus the conversion is beneficial and constructive only when we are executing it from other tools of Selenium Suite.

Step 1 – Click on the options tab under the menu bar and select the programming language format under format option in order to convert the test script into our desired format.


Gmail_Login.html - Selenium IDE 2.5.0 *
File Edit Actions Options Help
Base URL https://a
Fast Slow
Table Source
Options...
Format
Clipboard Format
Reset IDE Window
Clear history
HTML
Ruby / RSpec / WebDriver
Ruby / Test::Unit / WebDriver
Ruby / RSpec / Remote Control
Ruby / Test::Unit / Remote Control
Python 2 / unittest / WebDriver
Python 2 / unittest / Remote Control
Java / JUnit 4 / WebDriver
Java / JUnit 4 / WebDriver Backed
Java / JUnit 4 / Remote Control
Java / JUnit 3 / Remote Control
Java / TestNG / Remote Control
C# / NUnit / WebDriver
C# / NUnit / Remote Control
To convert the test script into desired format, Click Options ->Format -> (Programming Language)
HTML is the default format provided by Selenium IDE
Available Programming Language formats
Command
Target

Command	
open	/ServiceLogin?passive=120
assertTitle	Sign in - Google Accounts
type	id=Email
type	id=Passwd
clickAndWait	id=signIn

Step 2 – As soon as we select our Desired Programming language format ("Java / JUnit4 / WebDriver" in our case), a prompt box appears which says "Changing format is now marked experimental! If you continue, recording and playback may not work, your changes may be lost and you may have to copy and paste the test in a text editor to save. It is better to make a copy of your test cases before you continue. Do you still want to proceed?" Click "OK" to continue.


Gmail_Login.html - Selenium IDE 2.5.0 *
File Edit Actions Options Help
Base URL https://accounts.google.com/
Fast Slow
Table Source
Table view i.e. the HTML view is disabled and the source view is enabled
The converted java code

```
  private WebDriver driver;
  private String baseUrl;
  private boolean acceptNextAlert = true;
  private StringBuffer verificationErrors = new StringBuffer(

  @Before
  public void setUp() throws Exception {
    driver = new FirefoxDriver();
    baseUrl = "https://accounts.google.com/";
    driver.manage().timeouts().implicitlyWait(30, TimeUnit.SE
  }

  @Test
  public void testGmailLogin() throws Exception {
    driver.get(baseUrl + "/ServiceLogin?passive=1209600&conti
    assertEquals("Sign in - Google Accounts", driver.getTitle
```

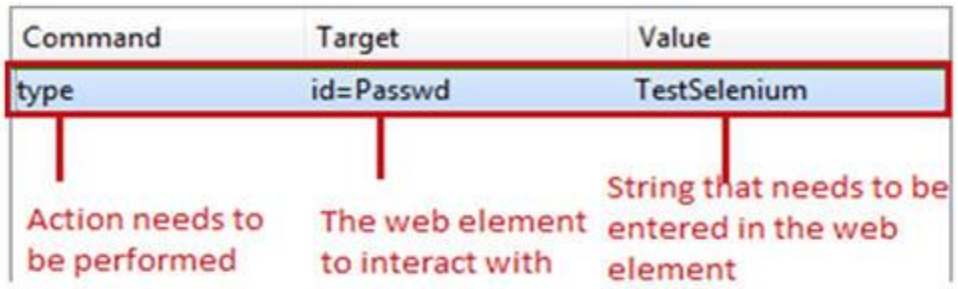
Thus, the above converted code can be executed by using WebDriver.

Mark that editing or modifying Selenium IDE test scripts from Source View is not advisable. If done so, the tool might introduce several repercussions. Several known bugs are already associated with it.

<u>**Selenium IDE Commands**</u>

Each Selenium IDE test step can chiefly be split into following three components:

- Command
- Target
- Value



Command	Target	Value
type	id=Passwd	TestSelenium

Action needs to be performed
The web element to interact with
String that needs to be entered in the web element

Types of Selenium IDE commands

There are three flavors of Selenium IDE commands. Each of the test step in Selenium IDE falls under any of the following category.

- Actions
- Accessors
- Assertions

<u>**Actions**</u>

Actions are those commands which interact directly with the application by either altering its state or by pouring some test data.

For Example, “type” command lets the user to interact directly with the web elements like text box. It allows them to enter a specific value in the text box and as when the value is entered; it is showed on the UI as well.

Another example is “click” command. “click” command lets the user to manipulate with the state of the application.

In case of failure of an action type command, the test script execution halts and rest of the test steps would not be executed.

<u>**Accessors**</u>

Accessors are those commands which allows user to store certain values to a user defined variable. These stored values can be later on used to create assertions and verifications.

For example, “storeAllLinks” reads and stores all the hyperlinks available within a web page into a user defined variable. Remember the variable is of array type if there are multiple values to store.

Assertions

Assertions are very similar to Accessors as they do not interact with the application directly. Assertions are used to verify the current state of the application with an expected state.

Forms of Assertions:

#1. assert – the “assert” command makes sure that the test execution is terminated in case of failure.

#2. verify – the “verify” command lets the Selenium IDE to carry on with the test script execution even if the verification is failed.

#3. waitFor – the “waitFor” command waits for a certain condition to be met before executing the next test step. The conditions are like page to be loaded, element to be present. It allows the test execution to proceed even if the condition is not met within the stipulated waiting period.

Commonly used Selenium IDE commands

Command	**Description**	**#Arguments**
open	Opens a specified URL in the browser.	1
assertTitle, VerifyTitle	Returns the current page title and compares it with the specified title	1
assertElementPresent, verifyElementPresent	Verify / Asserts the presence of an element on a web page.	1
assertTextPresent, verifyTextPresent	Verify / Asserts the presence of a text within the web page.	1
type, typeKeys, sendKeys	Enters a value (String) in the specified web element.	2
Click, clickAt, clickAndWait	Clicks on a specified web element within a web page.	1
waitForPageToLoad	Sleeps the execution and waits until the page is loaded completely.	1

Command	Description	#Arguments
waitForElement Present	Sleeps the execution and waits until the specified element is present	1
chooseOkOnNext Confirmation, chooseCancelOn NextConfirmation	Click on ”OK” or “Cancel” button when next confirmation box appears.	0

Conclusion

In this tutorial, we tried to make you acquainted with the creation of Selenium IDE scripts. We also briefed you about the usage of various Selenium features.

Here are the cruxes of this article.

- Test script in Selenium IDE can be created using Record and Playback feature.
- The script creation mechanism can be divided into 3 processes – **Recording, Playing back** and **Saving** the test script.
- Selenium IDE allows the user to execute a single test step within the test script without executing the entire test script. “**Execute this command**” is the option which makes this obtainable.
- User is leveraged to set the execution speed from the option within the toolbar.
- User can define any test step as a **Start point**. Thus, the execution will always initiate from that particular test step only.
- User can set multiple **Break points** to pause the execution at a certain test step.
- User can find and verify if the provided target value corresponds to the correct web element within the web page using **Find**
- Changing the source view to other formats is not recommended as there is a probability of loss of data.
- Remember to keep a copy of HTML test script before converting the test script into other non HTML formats.
- There are majorly three types of commands – **Actions, Accessors** and **Assertions**.
- Actions directly interact with the application and alter its state.
- Accessors are used to store an elements property in a user defined variable.
- Assertions are used to check if a specified condition is met or not.
- Assertions can further be categorized as **verify, assert** and **waitFor** commands**.**
- Verify makes sure that the test script execution is never halted if even if the verification fails.
- Assert lets no further execution of the test script in case of failure.
- WaitFor waits for a stipulated period for a certain condition to meet.
- **Some of the Selenium IDE commands which are used commonly are:**

- open
- assertTitle / VerifyTitle
- AssertForElementPresent / VerifyForElementPresent
- AssertForTextPresent / VerifyForTextPresent
- type / typeAndWait / sendKeys
- click /clickAt / clickAndWait
- waitForPageToLoad
- waitForElementPresent
- chooseOkOnNextConfirmation / chooseCancelOnNextConfirmation

Next Tutorial #4: There is another tool which plays a very important role in assisting us to create effective test scripts known as "Firebug". Firebug helps us in inspecting the properties of web elements and web pages. Thus the next tutorial is comprised of installation of Firebug and its usability. We would also **create a test script manually using firebug and Selenium IDE.**

Note: Do not miss the next tutorial because of its great importance with respect to our forthcoming tutorials on WebDriver.

As usual let us know your queries in comments below.

How to Use Firebug for Creating Selenium Scripts – Selenium Tutorial #4

In the previous tutorial, we learned how to create automated test scripts using Selenium IDE and its recording feature. We also flipped through populous features of Selenium IDE. We aimed at harbingering the reader with the most vital features and commands of Selenium IDE.

Just a reminder – this is our 4th tutorial in free Selenium training series.

Now that you are accustomed and capable of creating automated scripts using recording mode of Selenium IDE, let us move ahead with another tool which plays a very important role in assisting us to create effective test scripts known as "Firebug". Firebug helps us in inspecting the properties of web elements and web pages.

Thus, this tutorial is comprised of installation of Firebug and its usability.

Take a note that the content of this tutorial is not only applicable in context of Selenium IDE; rather it can be applied to each and every tool of Selenium suite. Thus I would preferably be using term Selenium instead of Selenium IDE.

In this tutorial lets learn how to use Firebug add-on for creating Selenium scripts. In the process we will also learn how to install Firebug.

Introduction to Firebug

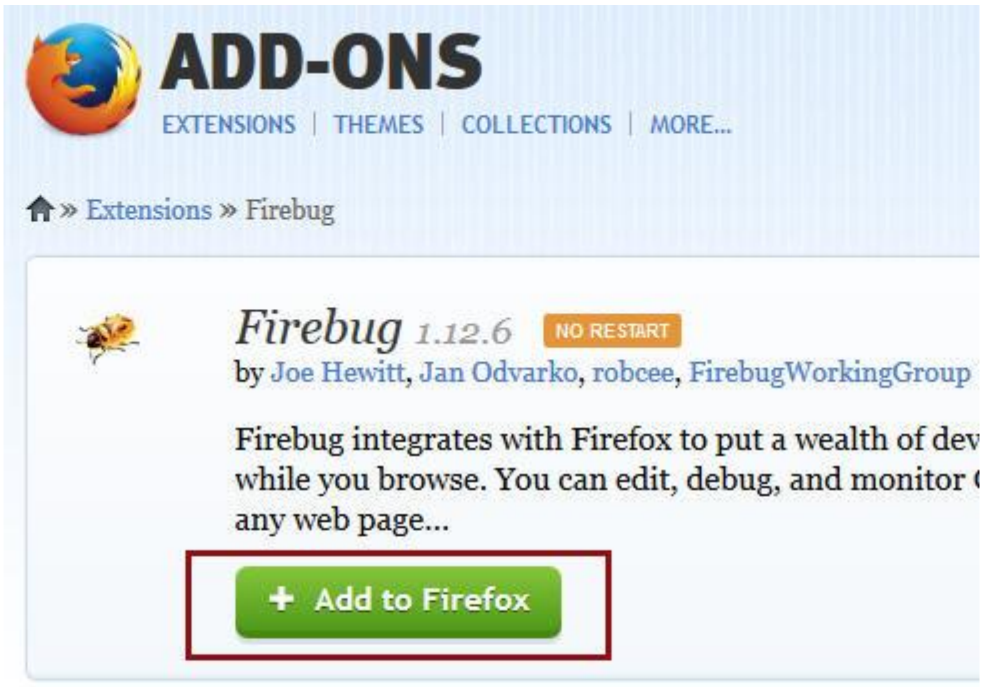
Firebug is a Mozilla Firefox add-on. This tool helps us in identifying or to be more particular inspecting HTML, CSS and JavaScript elements on a web page. It helps us identifying the elements uniquely on a webpage. The elements can be found uniquely based on their locator types which we would be discussing later in this tutorial.

How to Install Firebug?

For the ease of understanding, we would bifurcate the installation process into the following steps.

Step -1: Launch the Mozilla Firefox browser and navigate to this Firebug add-on download page. The URL takes us to Firefox add-ons section.

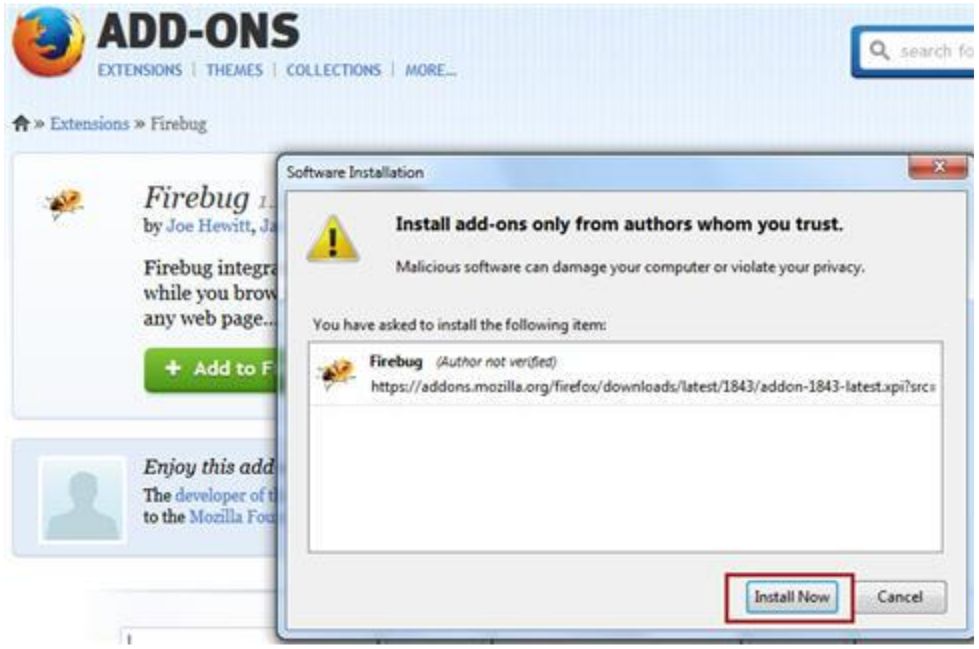
Step -2: Click on the "Add to Firefox" button present on the webpage. Refer the following figure for the same.


ADD-ONS
EXTENSIONS | THEMES | COLLECTIONS | MORE...
» Extensions » Firebug
Firebug 1.12.6 NO RESTART
by Joe Hewitt, Jan Odvarko, robcee, FirebugWorkingGroup
Firebug integrates with Firefox to put a wealth of dev
while you browse. You can edit, debug, and monitor
any web page...
+ Add to Firefox

Step-3: As soon as we click on the "Add to Firefox" button, a security alert box would appear, click on the "Allow" button now.

Step-4: Now Firefox downloads the add-on in the backdrop and a progress bar is displayed.

Step-5: As soon as the process completes, software installation window appears. Now click on the "Install Now" button.



Install add-ons only from authors whom you trust.

Malicious software can damage your computer or violate your privacy.

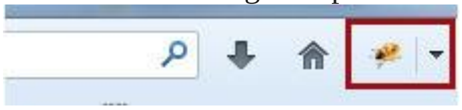
You have asked to install the following item:

Firebug (Author not verified)
https://addons.mozilla.org/firefox/downloads/latest/1843/addon-1843-latest.xpi?src=

Step-6: As soon as the installation completes, a pop up appears saying that the firebug has been installed successfully. Now choose to close this pop up.

Note: Unlike Selenium IDE, we are not required to restart the Firefox to reflect the firebug installation, rather it comes readily.

Step-7: Now to launch firebug, we can opt either of the following ways:

- Press F12
- Click on the firebug icon present in the extreme top-right corner of the Firefox window.
- 
- Click on Firefox menu bar -> Web Developer -> firebug -> Open Firebug.

Step-8: Now the firebug can be seen at the bottom of the Firefox window.

Now that we have downloaded and installed firebug, let's move ahead with the types of locators that we would be creating using firebug.

<u>Creating Selenium Script using Firebug</u>

Unlike Selenium IDE, In Firebug, we create automated test scripts manually by adding multiple test steps to form a logical and consistent test script.

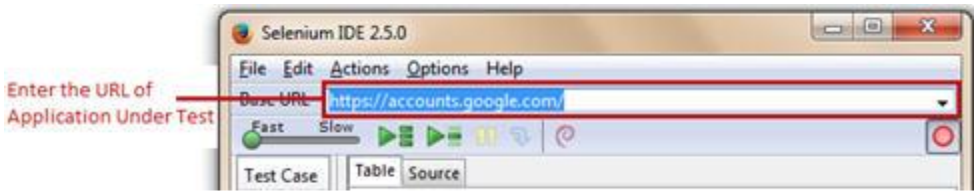
Let us follow a progressive approach and understand the process step by step.

Scenario:

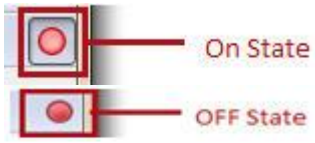
- Open "https://accounts.google.com".
- Assert Title of the application
- Enter an invalid username and invalid password and submit the details to login.

Step 1 – Launch the Firefox and open Selenium IDE from the menu bar.

Step 2 – Enter the address of application under test ("https://accounts.google.com") inside the Base URL textbox.

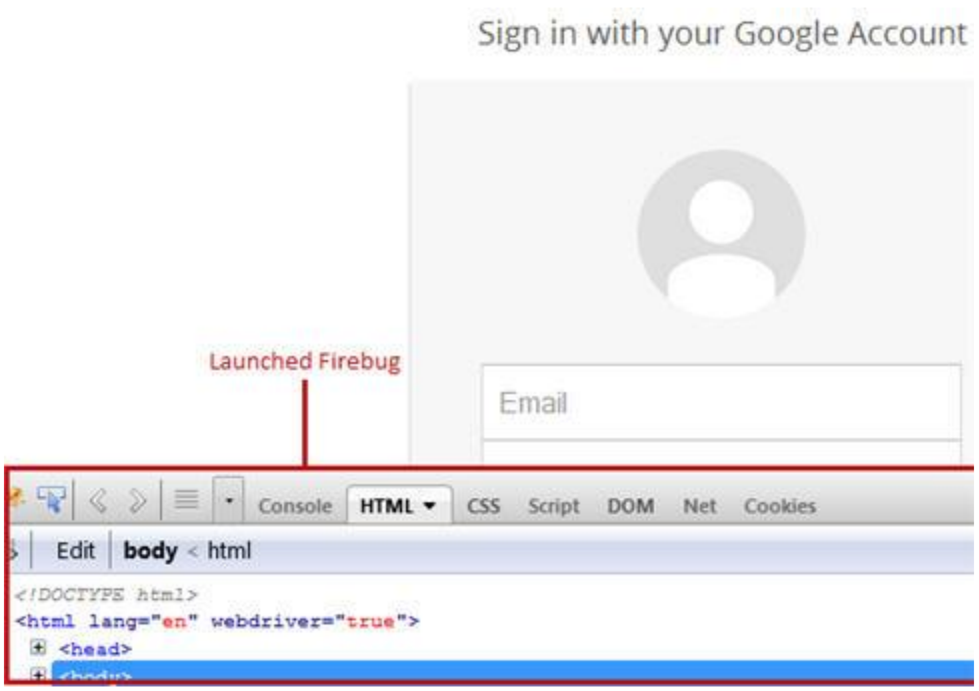

Selenium IDE 2.5.0
File Edit Actions Options Help
Enter the URL of Application Under Test
Base URL
https://accounts.google.com/
Fast Slow
Test Case
Table Source

Step 3 – By default, the Record button is in ON state. Remember to tune it OFF state so as to disable the recording mode. Notice if the recording mode is in ON state, it may result in recording our interactions with the web browser.



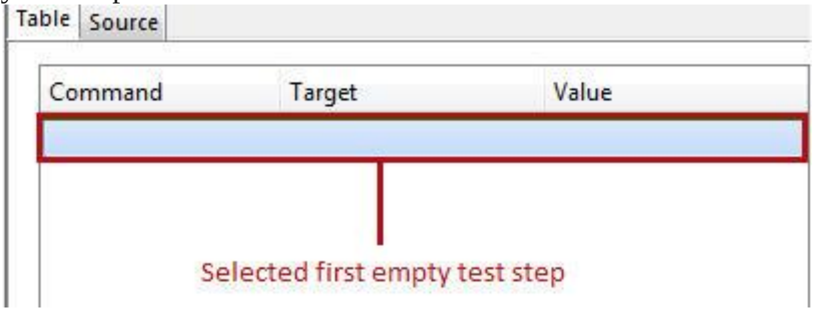
Step 4 – Open the application under test (https://accounts.google.com) in the Firefox.

Step 5 – Launch Firebug in the web browser.

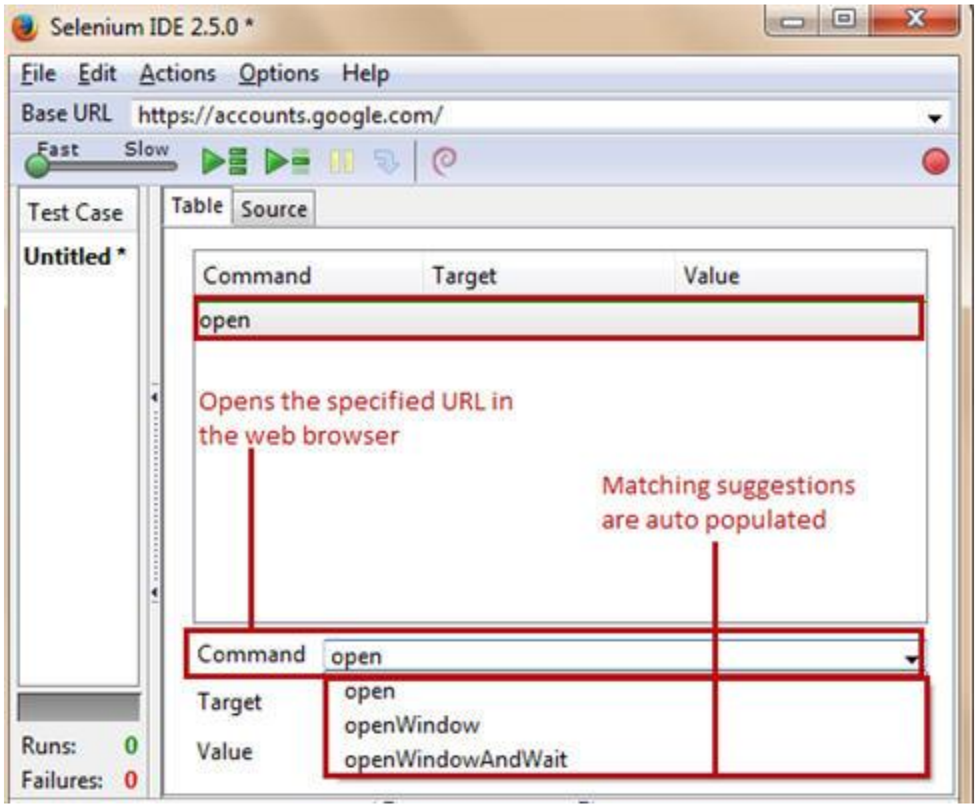


```
<!DOCTYPE html>
<html lang="en" webdriver="true">
  <head>
```

Step 6 – Select the empty test step within the Editor.



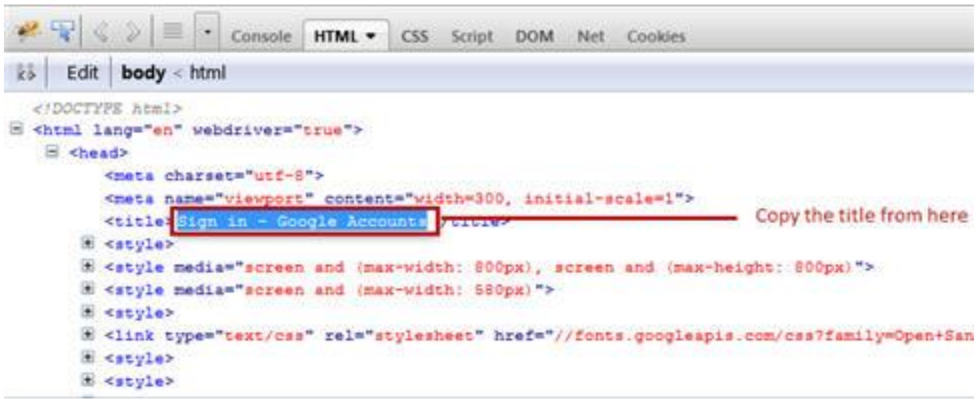
Step 7 – Type "open" in the command text box present in the Editor Pane. The "open" command opens the specified URL in the web browser.



Recommendation: While typing commands in the command text box, user can leverage the feature of auto selection. Thus, as soon as the user types a sequence of characters, the matching suggestions would be auto populated.

User can also click on the dropdown available within the command text box to look at all the commands provided by Selenium IDE.

Step 8 – Now, motion towards the Firebug section within the web browser, expand "head" section of the HTML code. Notice the HTML tag <title>. Thus to assert the title of the webpage, we would require the value of the <title> tag.

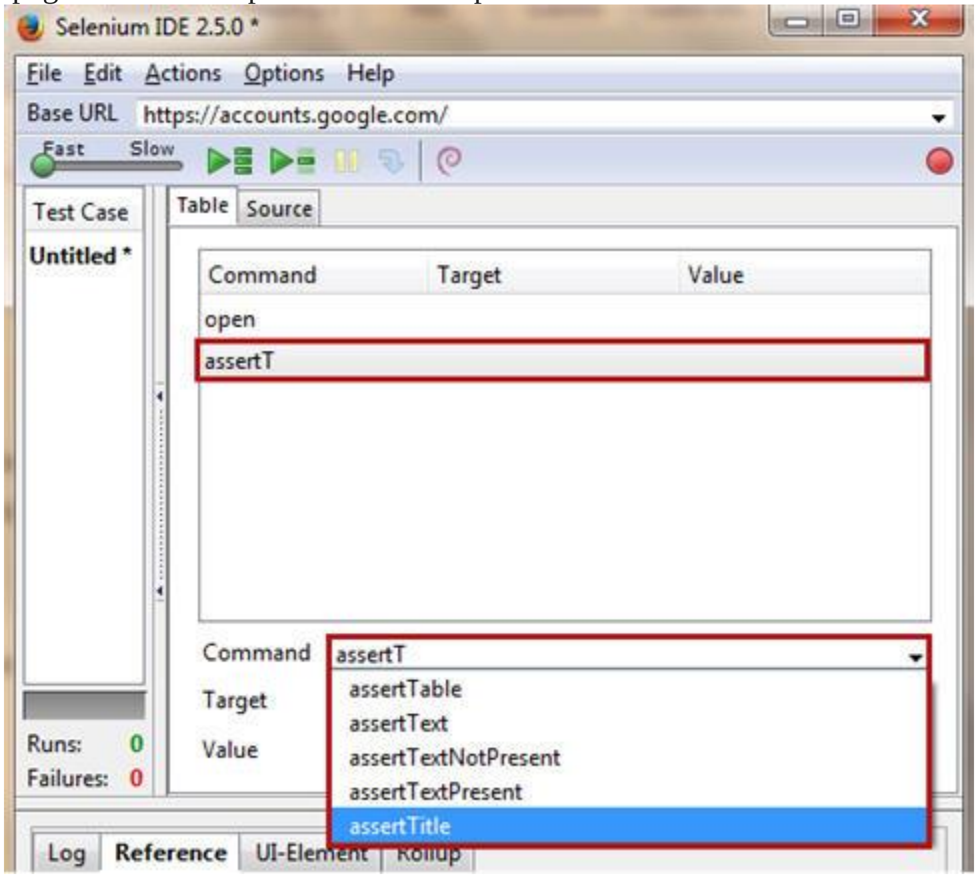


```
<!DOCTYPE html>
<html lang="en" webdriver="true">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=300, initial-scale=1">
    <title>Sign in - Google Accounts</title>
    <style>
    <style media="screen and (max-width: 800px), screen and (max-height: 800px)">
    <style media="screen and (max-width: 580px)">
    <style>
    <link type="text/css" rel="stylesheet" href="//fonts.googleapis.com/css?family=Open+San
    <style>
    <style>
```

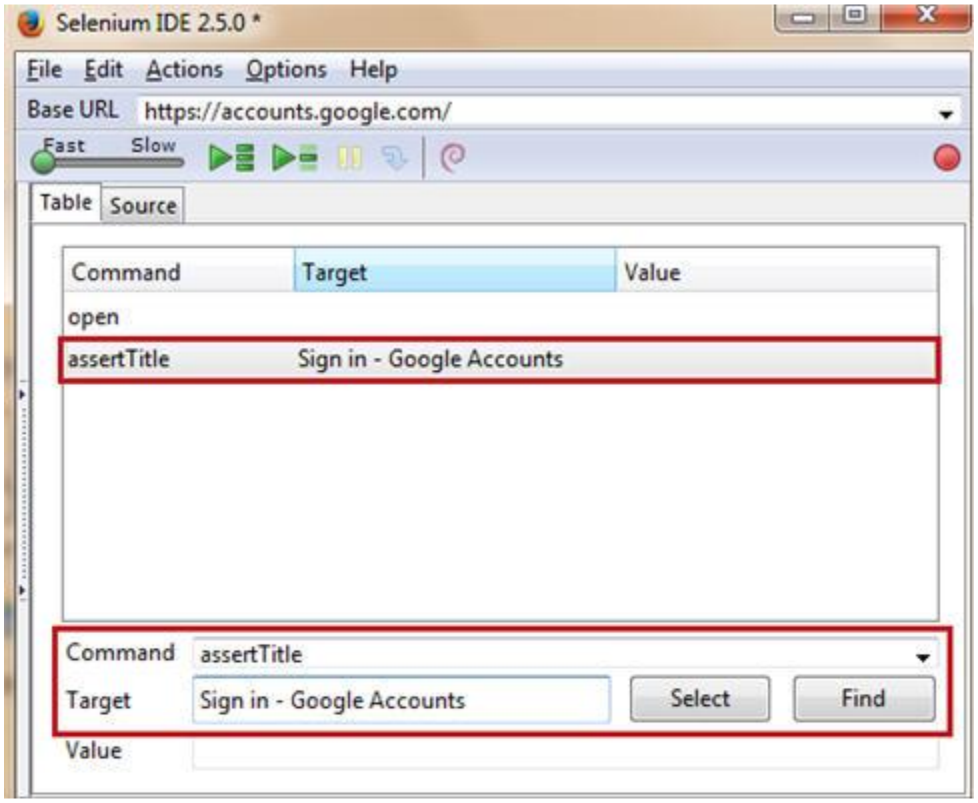
Copy the title of the webpage which is "Sign in – Google Accounts" in our case.

Step 9 – Select the second empty test step within the Editor.

Step 10 – Type “assertTitle” in the command text box present in the Editor Pane. The “assertTitle” command returns the current page title and compares it with the specified title.



Step 11 – Paste the title copied in step 8 into the Target field of the second.

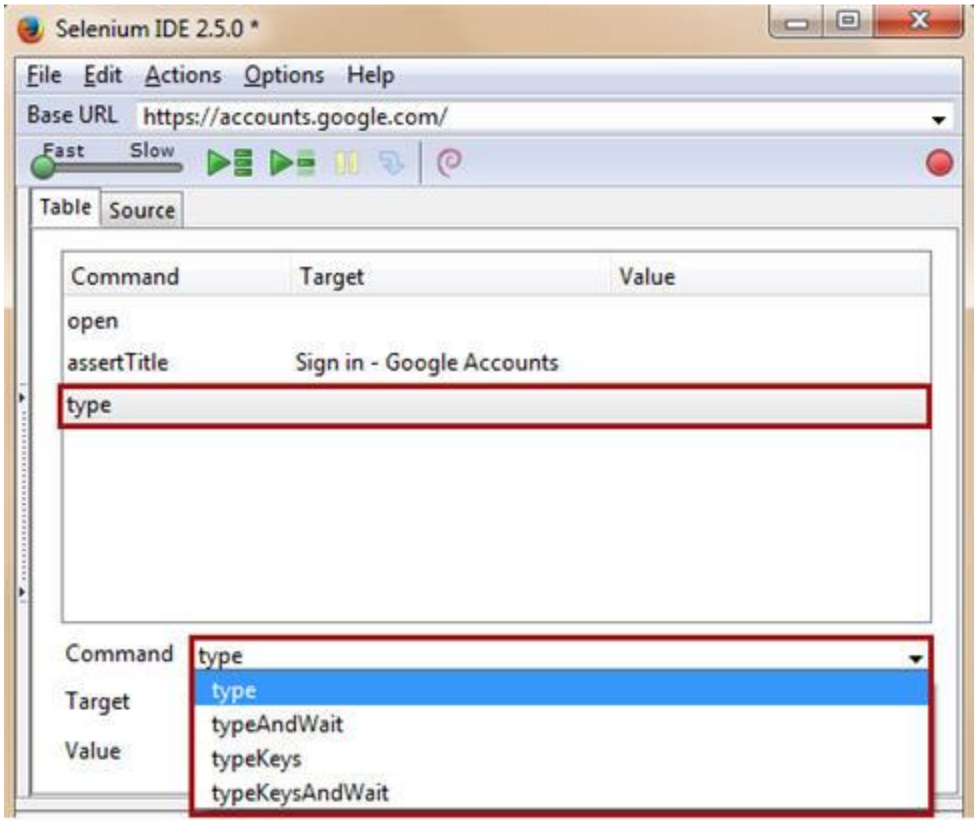

Selenium IDE 2.5.0 *
File Edit Actions Options Help
Base URL https://accounts.google.com/
Fast Slow
Table Source

Command	Target	Value
open		
assertTitle	Sign in - Google Accounts	

Command: assertTitle
Target: Sign in - Google Accounts
Value:

Step 12 – Now select the third empty test step in the Editor Pane

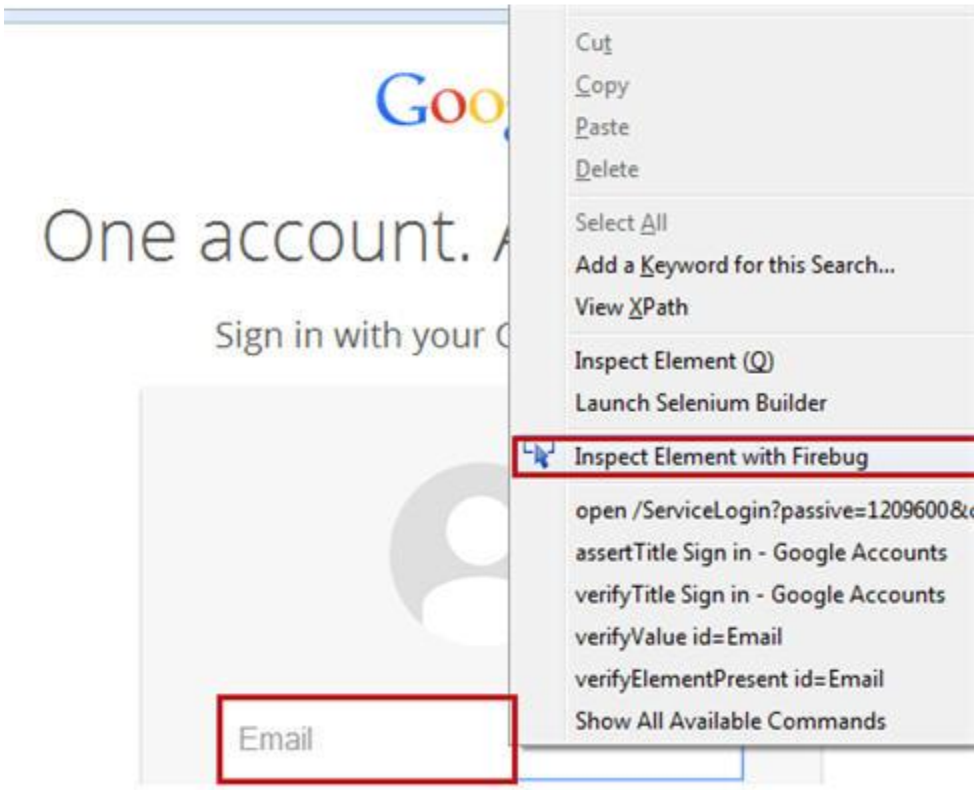
Step 13 – Type "type" command within the command text box. The "type" command enters a value in the specified web element on to the GUI.


Selenium IDE 2.5.0 *
File Edit Actions Options Help
Base URL https://accounts.google.com/
Fast Slow
Table Source

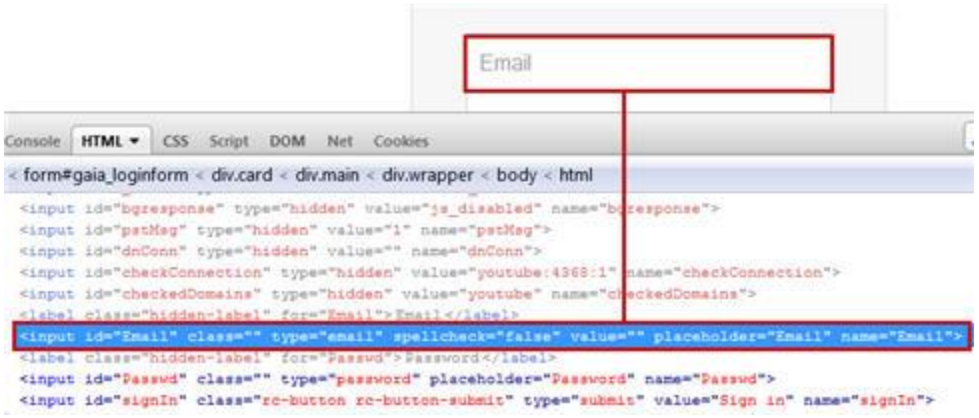
Command	Target	Value
open		
assertTitle	Sign in - Google Accounts	
type		

Command: type
Target
Value

Step 14 – Now switch to the web browser, bring the mouse cursor to the “Email” textbox within the login form and press a right click.


Goo
One account.
Sign in with your
Email
Cut
Copy
Paste
Delete
Select All
Add a Keyword for this Search...
View XPath
Inspect Element (Q)
Launch Selenium Builder
Inspect Element with Firebug
open /ServiceLogin?passive=1209600&
assertTitle Sign in - Google Accounts
verifyTitle Sign in - Google Accounts
verifyValue id=Email
verifyElementPresent id=Email
Show All Available Commands

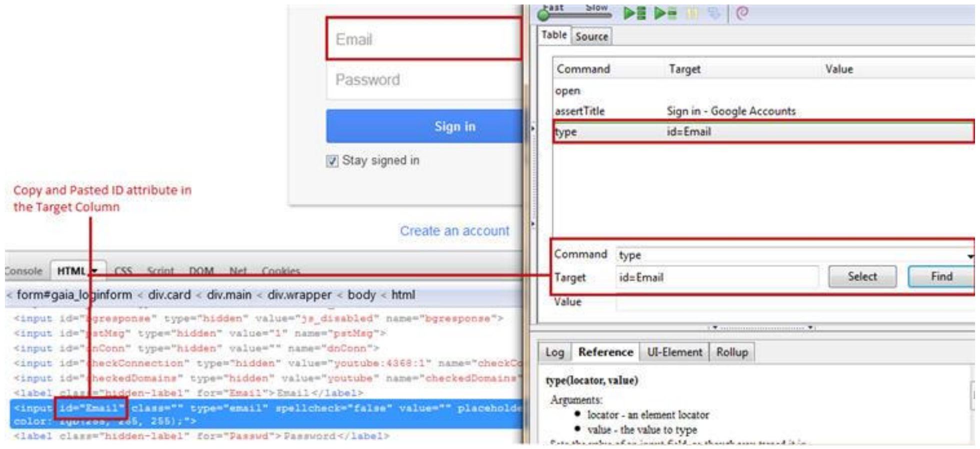
Choose “Inspect Element with Firebug”option. Notice that the Firebug automatically highlights the corresponding HTML code for the web element i.e. “Email Textbox”.


Email
Console
HTML
CSS
Script
DOM
Net
Cookies
< form#gaia_loginform < div.card < div.main < div.wrapper < body < html

```
<input id="bgresponse" type="hidden" value="js_disabled" name="bgresponse">
<input id="pstMsg" type="hidden" value="1" name="pstMsg">
<input id="dnConn" type="hidden" value="" name="dnConn">
<input id="checkConnection" type="hidden" value="youtube:4368:1" name="checkConnection">
<input id="checkedDomains" type="hidden" value="youtube" name="checkedDomains">
<label class="hidden-label" for="Email">Email</label>
<input id="Email" class="" type="email" spellcheck="false" value="" placeholder="Email" name="Email">
<label class="hidden-label" for="Passwd">Password</label>
<input id="Passwd" class="" type="password" placeholder="Password" name="Passwd">
<input id="signIn" class="rc-button rc-button-submit" type="submit" value="Sign in" name="signIn">
```

Step 15 – The HTML code in the above illustration manifests the distinct property attributes belonging to the “Email” text box. Notice that there are four properties (ID, type, placeholder and name) that uniquely identify the web element on the web page. Thus it’s up to the user to choose one or more than one property to identify the web element.

Thus, in this case, we choose ID as the locator. Copy the ID value and paste it in the Target field of the third test step prefixed with “id=” to indicate Selenium IDE to locate a web element having ID as “Email”.


Email
Password
Sign in
Stay signed in
Create an account
Copy and Pasted ID attribute in the Target Column
Console HTML CSS Script DOM Net Cookies
Fast Slow
Table Source
Select
Find
Log Reference UI-Element Rollup

< form#gaia_loginform < div.card < div.main < div.wrapper < body < html

```
<input id="bgresponse" type="hidden" value="js_disabled" name="bgresponse">
<input id="pstMsg" type="hidden" value="1" name="pstMsg">
<input id="dnConn" type="hidden" value="" name="dnConn">
<input id="checkConnection" type="hidden" value="youtube:4368:1" name="checkCo
<input id="checkedDomains" type="hidden" value="youtube" name="checkedDomains"
<label class="hidden-label" for="Email">Email</label>
<input id="Email" class="" type="email" spellcheck="false" value="" placeholde
color: rgb(255, 255, 255);">
<label class="hidden-label" for="Passwd">Password</label>
```

Command	Target	Value
open		
assertTitle	Sign in - Google Accounts	
type	id=Email	

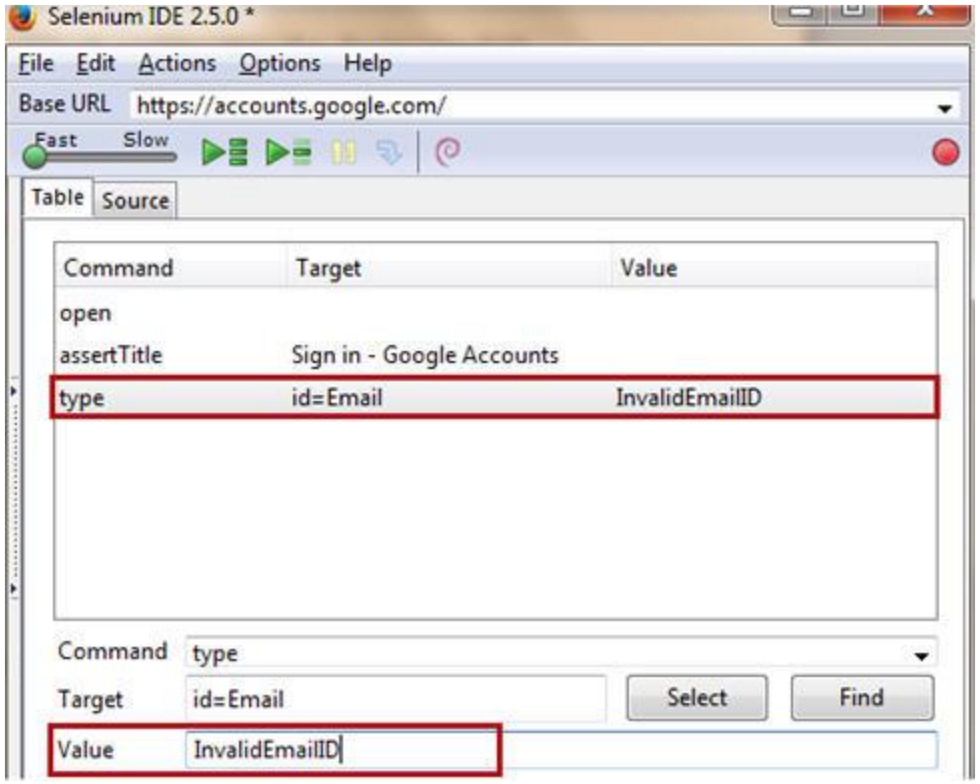
Command: type
Target: id=Email
Value:

type(locator, value)
Arguments:
- locator - an element locator
- value - the value to type

Make a note that Selenium IDE is case sensitive, thus type the attribute value carefully and precisely the same as it is displayed in the HTML code.

Step 16 – Click on the Find button to verify if the locator selected finds and locates the designated UI element on the web page.

Step 17 – Now, the next step is to enter the test data in the Value textbox of the third test step within the Editor Pane. Enter "InvalidEmailID" in the Value textbox. User can alter the test data as and when it is desired.



Base URL: https://accounts.google.com/

Command	Target	Value
open		
assertTitle	Sign in - Google Accounts	
type	id=Email	InvalidEmailID

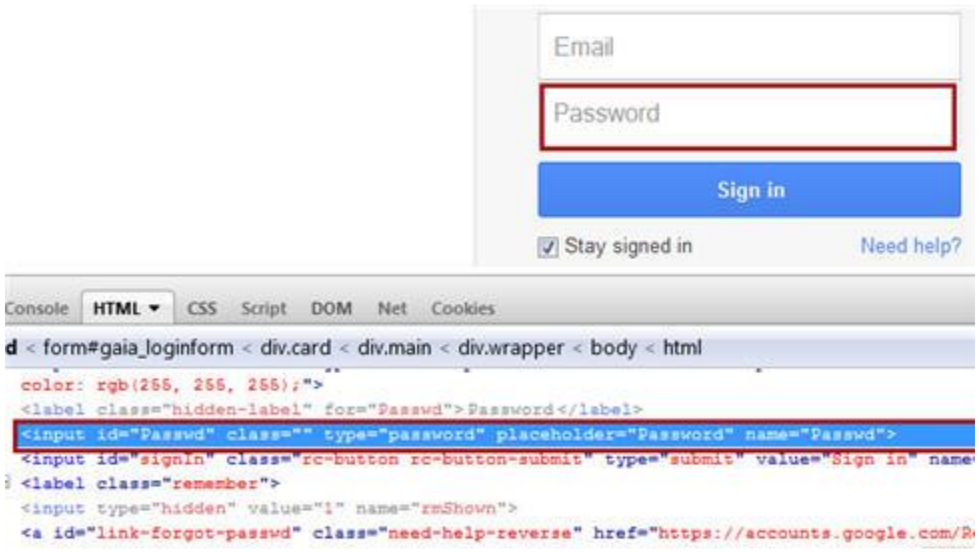
Command: type
Target: id=Email
Value: InvalidEmailID

Step 18 – Now select the fourth empty test step in the Editor Pane

Step 19 – Type "type" command within the command text box.

Step 20 – Now switch to the web browser, bring the mouse cursor to the "Password" textbox within the login form and press a right click.

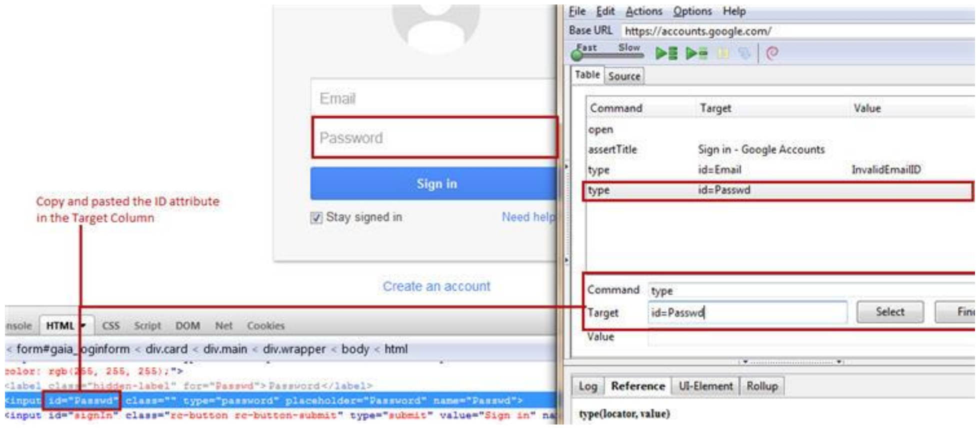
Choose “Inspect Element with Firebug”option.



```
color: rgb(255, 255, 255);">
<label class="hidden-label" for="Passwd">Password</label>
<input id="Passwd" class="" type="password" placeholder="Password" name="Passwd">
<input id="signIn" class="rc-button rc-button-submit" type="submit" value="Sign in" name
<label class="remember">
<input type="hidden" value="1" name="rmShown">
<a id="link-forgot-passwd" class="need-help-reverse" href="https://accounts.google.com/R
```

Step 21 – The HTML code below manifests the distinct property attributes belonging to the “Password” text box. Notice that there are four properties (ID, type, placeholder and name) that uniquely identify the web element on the web page. Thus it’s up to the user to choose one or more than one property to identify the web element.

Thus, in this case, we choose ID as the locator. Copy the ID value and paste it in the Target field of the third test step prefixed with “id=”.



Base URL https://accounts.google.com/

Command	Target	Value
open		
assertTitle	Sign in - Google Accounts	
type	id=Email	InvalidEmailID
type	id=Passwd	

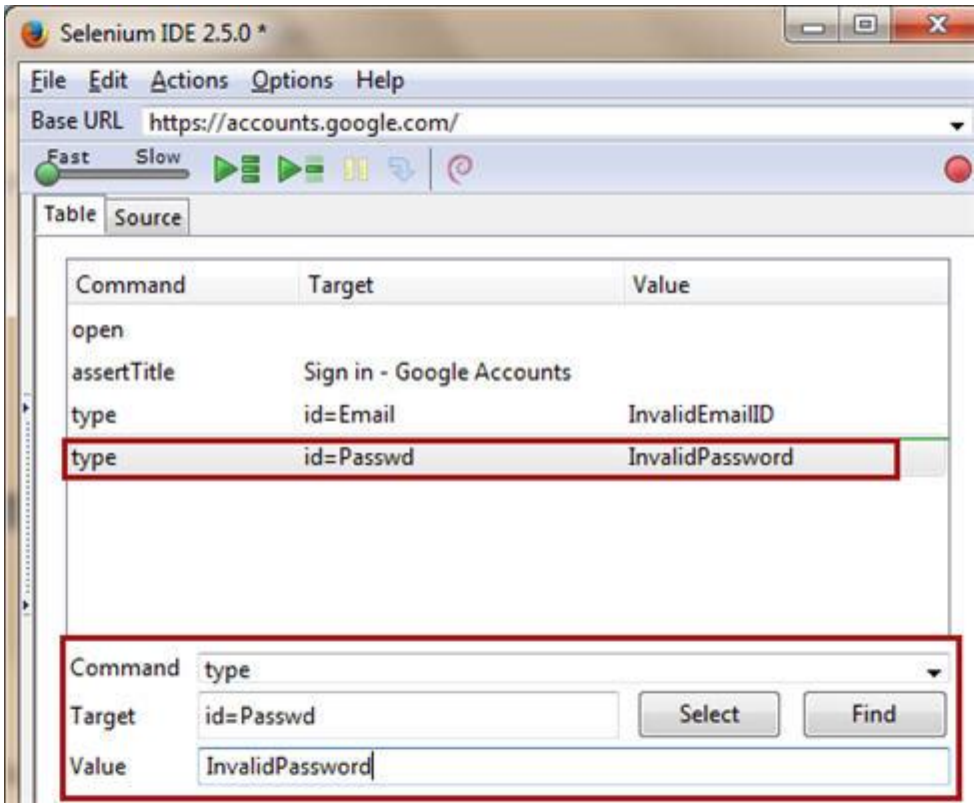
Command: type
Target: id=Passwd
Value:

type(locator, value)

```
color: rgb(255, 255, 255);">
<label class="hidden-label" for="Passwd">Password</label>
<input id="Passwd" class="" type="password" placeholder="Password" name="Passwd">
<input id="signIn" class="rc-button rc-button-submit" type="submit" value="Sign in" na
```

Step 22 – Click on the Find button to verify if the locator tabbed finds and locates the designated UI element on the web page.

Step 23 – Now, the next step is to enter the test data in the Value textbox of the fourth test step within the Editor Pane. Enter “InvalidPassword” in the Value textbox. User can alter the test data as and when it is desired.



Base URL https://accounts.google.com/

Command	Target	Value
open		
assertTitle	Sign in - Google Accounts	
type	id=Email	InvalidEmailID
type	id=Passwd	InvalidPassword

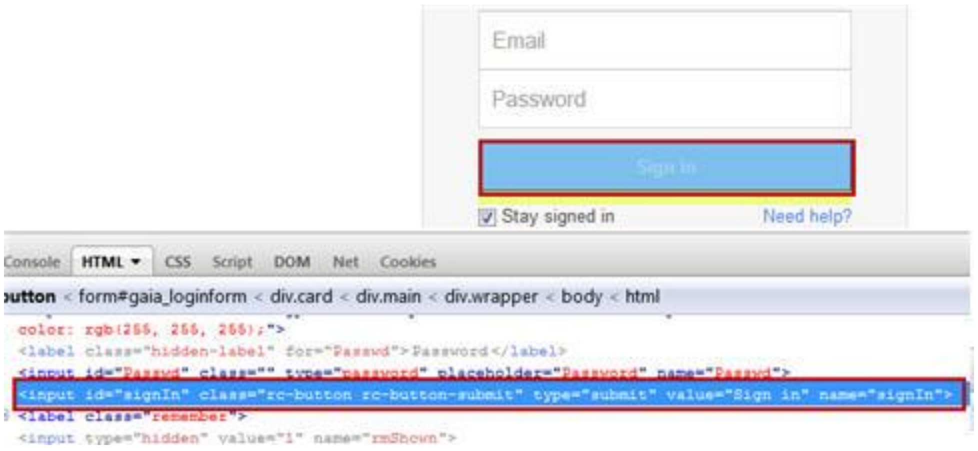
Command	type		
Target	id=Passwd	Select	Find
Value	InvalidPassword		

Step 24 – Now select the fifth empty test step in the Editor Pane

Step 25 – Type “click” command within the command text box. The “click” command clicks on a specified web element within the web page.

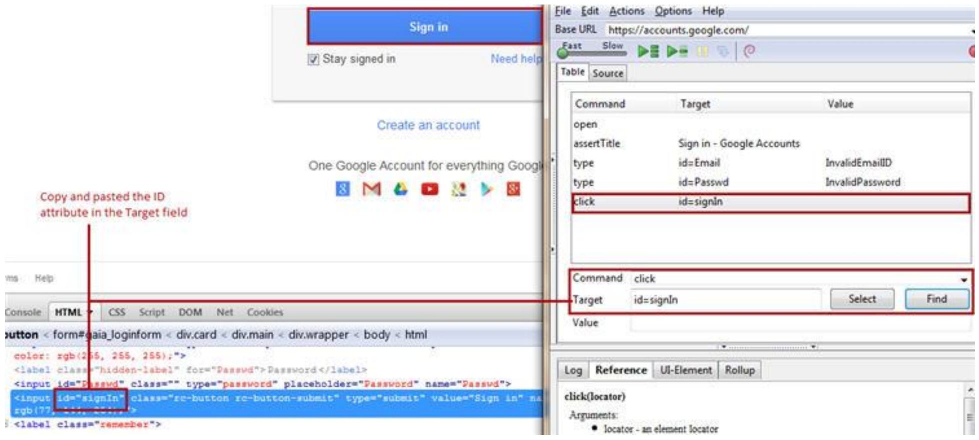
Step 26 – Now switch to the web browser, bring the mouse cursor to the “Sign in” button within the login form and press a right click.

Choose “Inspect Element with Firebug”option.



```
button < form#gaia_loginform < div.card < div.main < div.wrapper < body < html
color: rgb(255, 255, 255);">
<label class="hidden-label" for="Passwd">Password</label>
<input id="Passwd" class="" type="password" placeholder="Password" name="Passwd">
<input id="signIn" class="rc-button rc-button-submit" type="submit" value="Sign in" name="signIn">
<label class="remember">
<input type="hidden" value="1" name="rmShown">
```

Step 27 – The HTML code below manifests the distinct property attributes belonging to the “Sign in” button. Choose ID as the locator. Copy the ID value and paste it in the Target field of the third test step prefixed with “id=”.


Sign in
Stay signed in
Need help
Create an account
One Google Account for everything Google
Copy and pasted the ID attribute in the Target field
Console HTML CSS Script DOM Net Cookies
File Edit Actions Options Help
Fast Slow
Table Source
Select
Find
Log Reference UI-Element Rollup

Base URL https://accounts.google.com/

Command	Target	Value
open		
assertTitle	Sign in - Google Accounts	
type	id=Email	InvalidEmailID
type	id=Passwd	InvalidPassword
click	id=signIn	

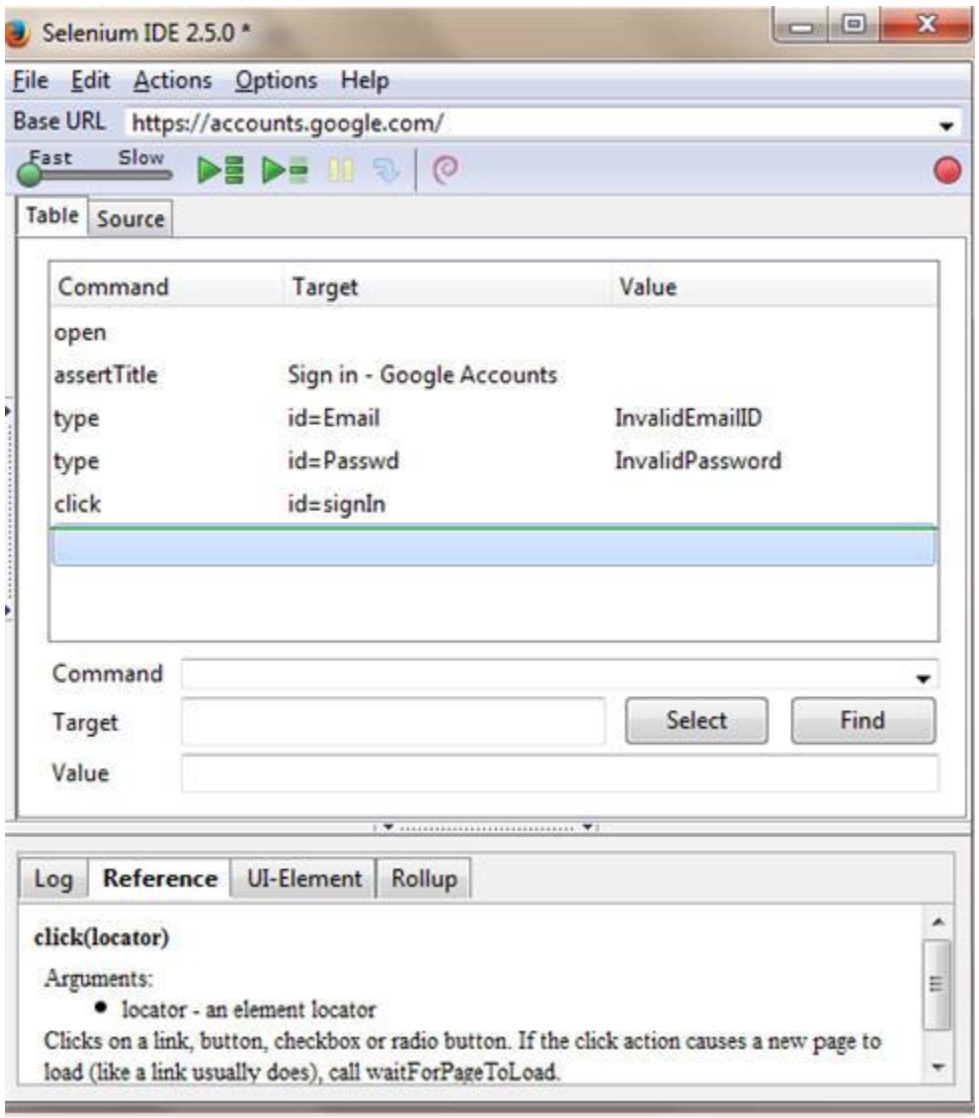
Command: click
Target: id=signIn
Value:

```
button < form#gaia_loginform < div.card < div.main < div.wrapper < body < html
color: rgb(255, 255, 255);">
<label class="hidden-label" for="Passwd">Password</label>
<input id="Passwd" class="" type="password" placeholder="Password" name="Passwd">
<input id="signIn" class="rc-button rc-button-submit" type="submit" value="Sign in" na
rgb(77, 144, 254);">
<label class="remember">
```

click(locator)
Arguments:
- locator - an element locator

Step 28 – Click on the Find button to verify if the locator picked finds and locates the designated UI element on the web page.

The test script is completed now. Refer the following illustration to view the finished test script.


Selenium IDE 2.5.0 *
File Edit Actions Options Help
Fast Slow
Table Source
Select
Find
Log Reference UI-Element Rollup

Base URL https://accounts.google.com/

Command	Target	Value
open		
assertTitle	Sign in - Google Accounts	
type	id=Email	InvalidEmailID
type	id=Passwd	InvalidPassword
click	id=signIn	

Command:
Target:
Value:

click(locator)
Arguments:
- locator - an element locator

Clicks on a link, button, checkbox or radio button. If the click action causes a new page to load (like a link usually does), call waitForPageToLoad.

Step 29 – Play back the created test script and Save it in the same way we did in the previous tutorial.

Conclusion

In this tutorial, we introduced yet another script creation tool or rather a tool that aids to script creation.

Firebug surprisingly has a great potential to locate web elements on a web page. Thus the user can leverage the tool’s capabilities in creating effective and efficient automation test scripts manually.

Next Tutorial #5: Moving ahead in the next tutorial, we would have a look at the **various types of locators in Selenium and their accessibility technique to build test scripts**. In the meantime reader can start building his/her automation test scripts using Firebug.
Have you used Firebug for inspecting HTML elements or for creating scripts? Do you find it useful? ***Please share your experience in comments***

How to Identify Web Elements Using Selenium Xpath and Other Locators – Selenium Tutorial #5

In the previous tutorial, we introduced you with another automation testing tool named as Firebug. We also created our own automation script manually using Firebug and its capabilities. We also learned to affix desired modifications into our script.

Moving ahead, in this tutorial we would have a look at the **various types of locators in Selenium and their accessibility technique to build test scripts**. Thus this tutorial is comprised of the detailed introduction to various types of locators.

This is our 5th tutorial in Selenium Tutorial series.

What is Locator?

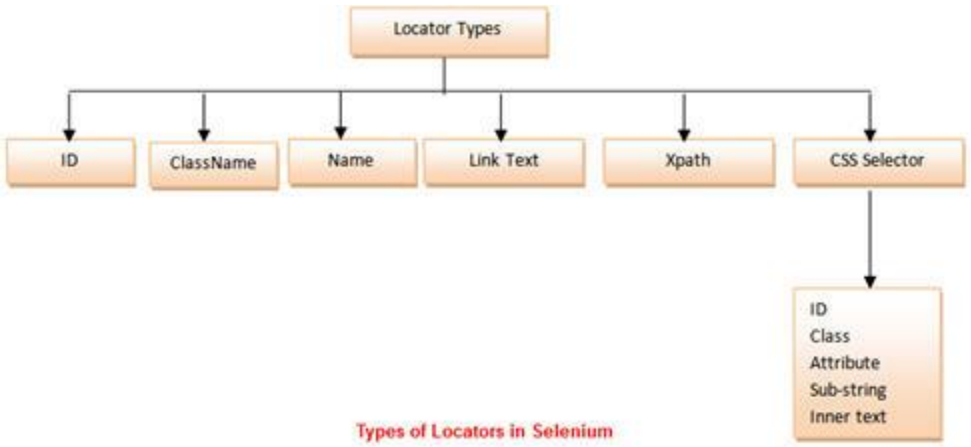
Locator can be termed as an address that identifies a web element uniquely within the webpage. Locators are the HTML properties of a web element which tells the Selenium about the web element it need to perform action on.

There is a diverse range of web elements. **The most common amongst them are:**

- Text box
- Button
- Drop Down
- Hyperlink
- Check Box
- Radio Button

Types of Locators

Identifying these elements has always been a very tricky subject and thus it requires an accurate and effective approach. Thereby, we can assert that more effective the locator, more stable will be the automation script. Essentially every Selenium command requires locators to find the web elements. Thus, to identify these web elements accurately and precisely we have different types of locators.



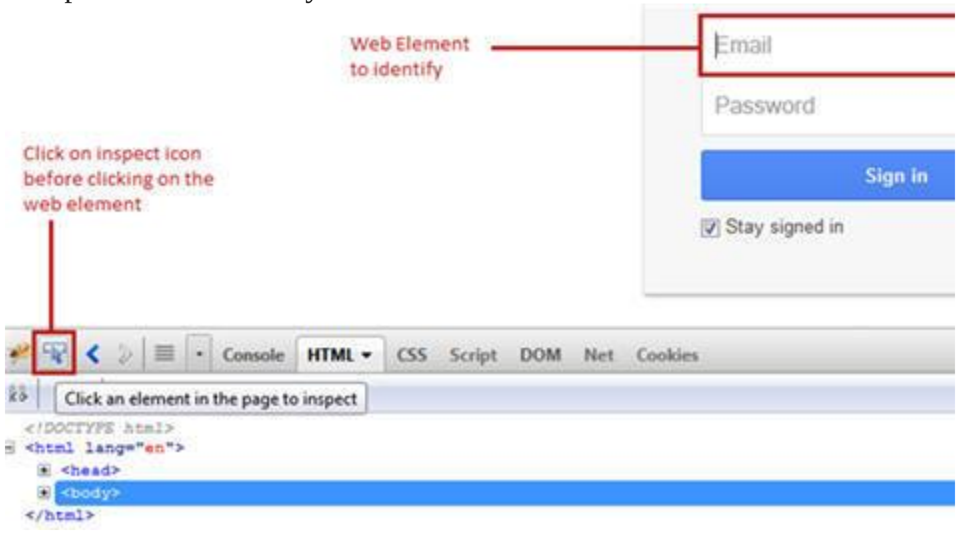
Now let's understand further by exercising each of them independently.
Before we start with the locators, let me take a moment to introduce the application under test. We would be using "https://accounts.google.com/" for locating different types of web elements using different locator types.

Using ID as a Locator
The best and the most popular method to identify web element is to use ID. The ID of an each element is alleged to be unique.

Web Element
1
1
ID

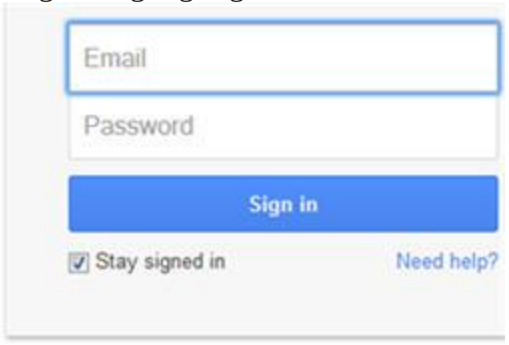
In this sample, we would access "Email" text box present in the login form at gmail.com.

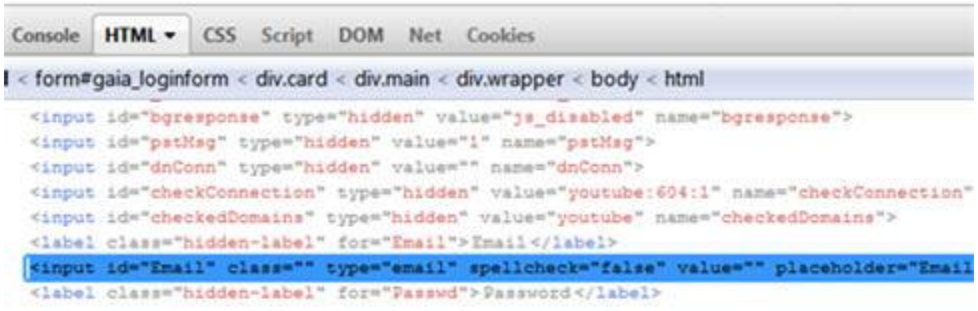
<u>Finding an ID of a web element using Firebug</u>
Step 1: Launch the web browser (Firefox) and navigate to "https://accounts.google.com/".
Step 2: Open firebug (either by pressing F12 or via tools).
Step 3: Click on the inspect icon to identify the web element.


Web Element to identify
Email
Password
Sign in
Stay signed in
Click on inspect icon before clicking on the web element
Console
HTML
CSS
Script
DOM
Net
Cookies
Click an element in the page to inspect

```
<!DOCTYPE html>
<html lang="en">
  <head>
  <body>
</html>
```

Step 4: Hover on the web element (Email textbox in our case) on which we desire to perform some action. In the firebug section, one can see the corresponding html tags being highlighted.





```
< form#gaia_loginform < div.card < div.main < div.wrapper < body < html
<input id="bgresponse" type="hidden" value="js_disabled" name="bgresponse">
<input id="pstMsg" type="hidden" value="1" name="pstMsg">
<input id="dnConn" type="hidden" value="" name="dnConn">
<input id="checkConnection" type="hidden" value="youtube:604:1" name="checkConnection"
<input id="checkedDomains" type="hidden" value="youtube" name="checkedDomains">
<label class="hidden-label" for="Email">Email</label>
<input id="Email" class="" type="email" spellcheck="false" value="" placeholder="Email
<label class="hidden-label" for="Passwd">Password</label>
```

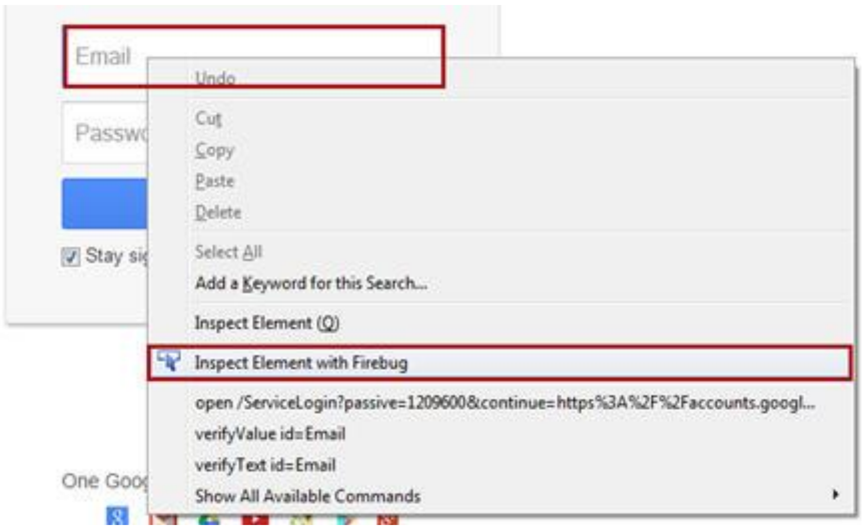
Step 5: Be cognizant about the ID attribute and take a note of it. Now we need to verify if the ID indentified is able to find the element uniquely and flawlessly.

Syntax: id = id of the element

In our case, the id is “Email”.

Alternative approach:

Instead of following step 2 to 4, we can directly locate / inspect the web element by right clicking on the web element (Email Textbox) whose locator value we need to inspect and clicking on the option “Inspect Element with Firebug”. Thus, this click event triggers the expansion of firebug section and the corresponding html tag would be highlighted.


Email
Passw
Stay si
One Goo
Undo
Cut
Copy
Paste
Delete
Select All
Add a Keyword for this Search...
Inspect Element (Q)
Inspect Element with Firebug
open /ServiceLogin?passive=1209600&continue=https%3A%2F%2Faccounts.googl...
verifyValue id=Email
verifyText id=Email
Show All Available Commands

Verify the locator value

Assuming that the browser is open and is re-directed to “https://accounts.google.com/”.

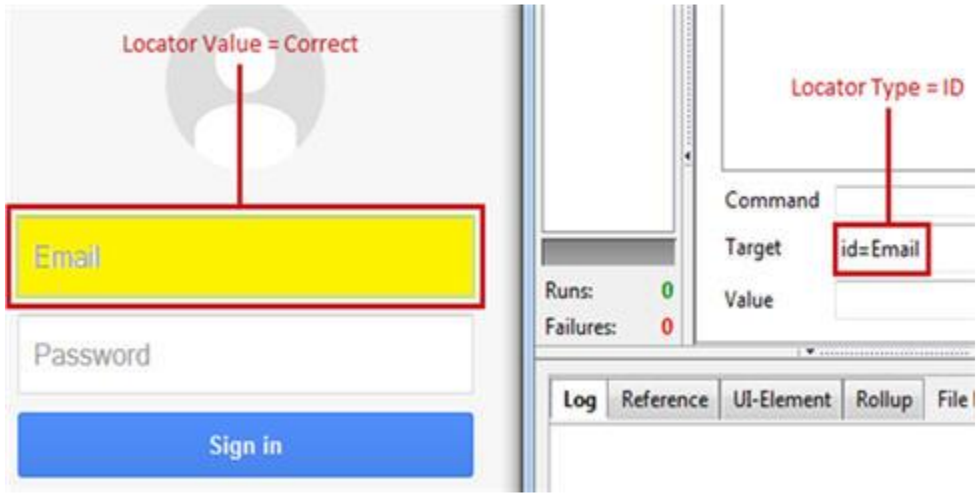
Step 1: Launch Selenium IDE.

Step 2: Click on the first row in the editor section.

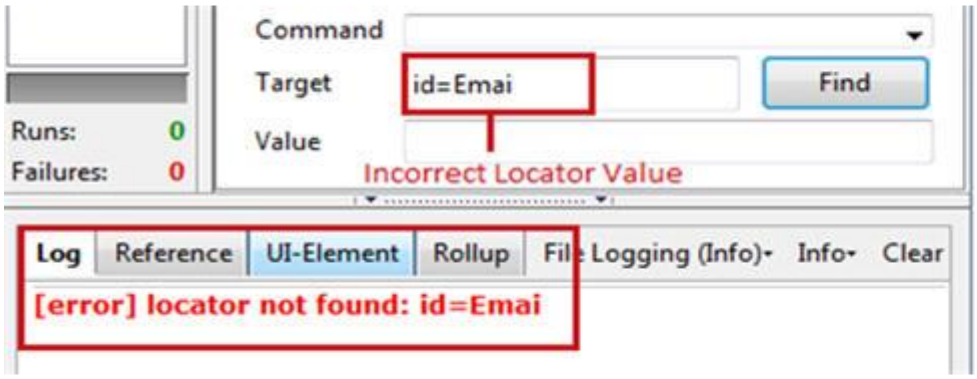
Step 3: Type “id=Email” i.e. the locator value in the target box.

Step 4: Click on the Find Button. If the provided locator value is legitimate then the Email textbox will be highlighted with yellow color with a florescent green border around the field. If the locator value provided is incorrect, an error message would be printed in the log pane at the bottom of Selenium IDE.

Case 1 – Locator Value = Correct



Case 2 – Locator Value = Incorrect



id=Emai

[error] locator not found: id=Emai

Step 5: In order to verify further, user can also execute “type” command against the given target by providing some value in the “Value” field. If the execution of the command enters the specified value in the Email text box that means the identified locator type is correct and accessible.

Using ClassName as a Locator

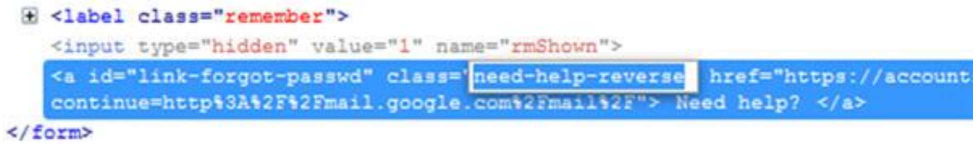
There is only a subtle difference between using ID as a locator and using classname as a locator.

In this sample, we would access “Need Help?” hyperlink present at the bottom of the login form at gmail.com.

<u>Finding a classname of a web element using Firebug</u>

Step 1: Locate / inspect the web element (“Need help?” link in our case) by right clicking on the web element whose locator value we need to inspect and clicking on the option “Inspect Element with Firebug”.

Step 2: Be cognizant about the classname attribute and take a note of it. Now we need to verify if the classname indentified is able to find the element uniquely and accurately.



```
<label class="remember">
  <input type="hidden" value="1" name="rmShown">
  <a id="link-forgot-passwd" class="need-help-reverse" href="https://account
  continue=http%3A%2F%2Fmail.google.com%2Fmail%2F"> Need help? </a>
</form>
```

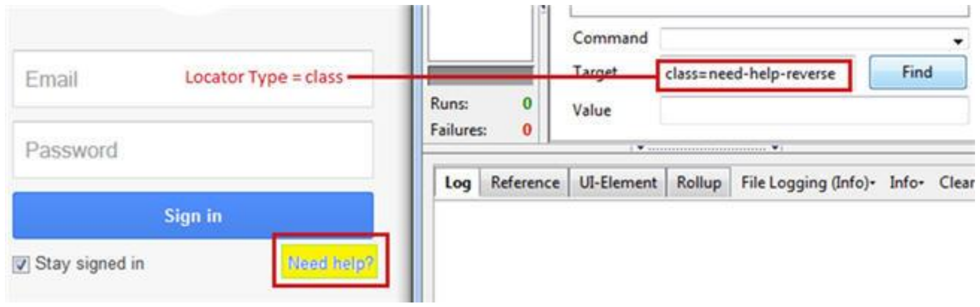
<u>Syntax</u>: class = classname of the element

In our case, the classname is “need-help-reverse”

<u>Verify the locator value</u>

Step 1: Type “class= need-help-reverse” in the target box in the Selenium IDE.

Step 2: Click on the Find Button. Notice that the hyperlink will be highlighted with yellow color with a florescent green border around the field.



class=need-help-reverse

Using name as a Locator

Locating a web element using name is very much analogous to previous two locator types. The only difference lies in the syntax.

In this sample, we would access "Password" text box present in the login form at gmail.com.

<u>**Syntax:**</u> name = name of the element

In our case, the name is "Passwd".

Verify the locator value

Step 1: Type "name= Passwd" in the target box and click on the Find Button. Notice that the "Password" textbox would be highlighted.

Using Link Text as a Locator

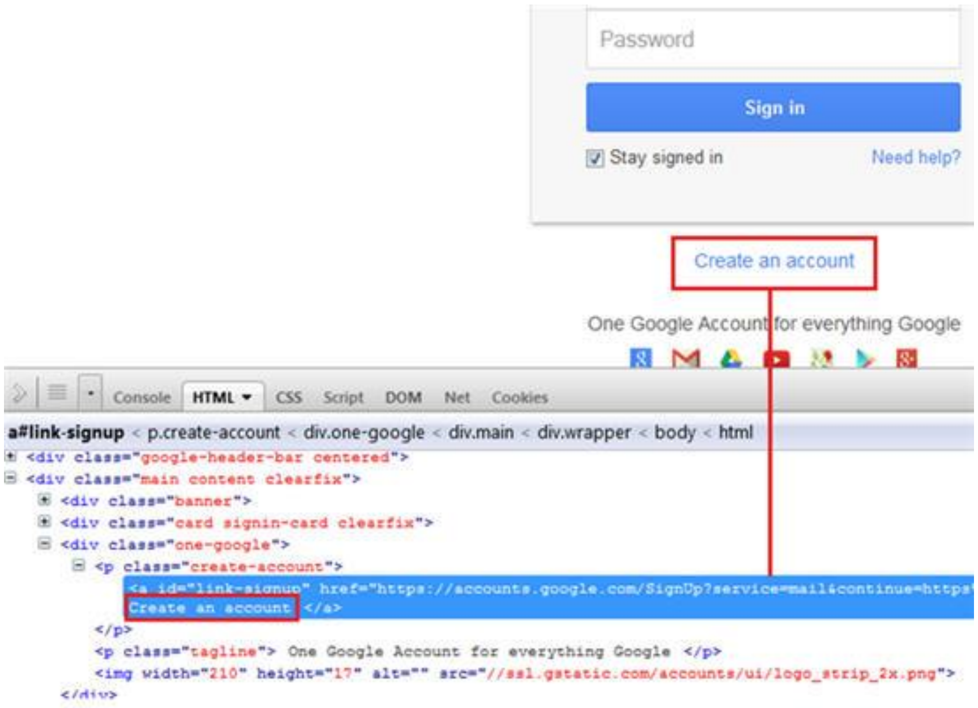
All the hyperlinks on a web page can be indentified using Link Text. The links on a web page can be determined with the help of anchor tag (<a>). The anchor tag is used to create the hyperlinks on a web page and the text between opening and closing of anchor tags constitutes the link text (<a>Some Text</a>).

In this sample, we would access "Create an account" link present at the bottom of the login form at gmail.com.

<u>**Finding a link text of a web element using Firebug**</u>

Step 1: Locate / inspect the web element ("Create an account" link in our case) by right clicking on the web element whose locator value we need to inspect and clicking on the option "Inspect Element with Firebug".

Step 2: Be cognizant about the text present within the <a> </a> tags and take a note of it. Hence this text will be used to identify the link on a web page uniquely.


Password
Sign in
Stay signed in
Need help?
Create an account
One Google Account for everything Google
Console
HTML
CSS
Script
DOM
Net
Cookies
a#link-signup < p.create-account < div.one-google < div.main < div.wrapper < body < html

```
<div class="google-header-bar centered">
<div class="main content clearfix">
    <div class="banner">
    <div class="card signin-card clearfix">
    <div class="one-google">
        <p class="create-account">
            <a id="link-signup" href="https://accounts.google.com/SignUp?service=mail&continue=https
            Create an account </a>
        </p>
        <p class="tagline"> One Google Account for everything Google </p>
        <img width="210" height="17" alt="" src="//ssl.gstatic.com/accounts/ui/logo_strip_2x.png">
    </div>
```

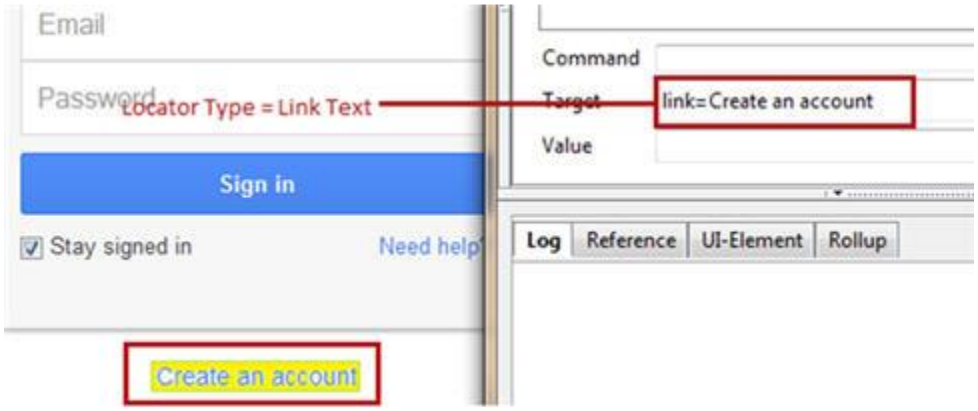
<u>**Syntax**</u>**:** link = link text of the element

In our case, the link text is "Create an account".

<u>**Verify the locator value**</u>

Step 1: Type "link=Create an account" i.e. the locator value in the target box in Selenium IDE.

Step 2: Click on the Find Button. Notice that the link would be highlighted with yellow color with a florescent green border around the field.



Using Xpath as a Locator

Xpath is used to locate a web element based on its XML path. XML stands for Extensible Markup Language and is used to store, organize and transport arbitrary data. It stores data in a key-value pair which is very much similar to HTML tags. Both being mark up languages and since they fall under the same umbrella, xpath can be used to locate HTML elements.

The fundamental behind locating elements using Xpath is the traversing between various elements across the entire page and thus enabling a user to find an element with the reference of another element.

Xpath can be created in two ways:

Relative Xpath

Relative Xpath begins from the current location and is prefixed with a "//".

For example: //span[@class='Email']

Absolute Xpath

Absolute Xpath begins with a root path and is prefixed with a "/".

For example: /html/body/div/div[@id='Email']

Key Points:

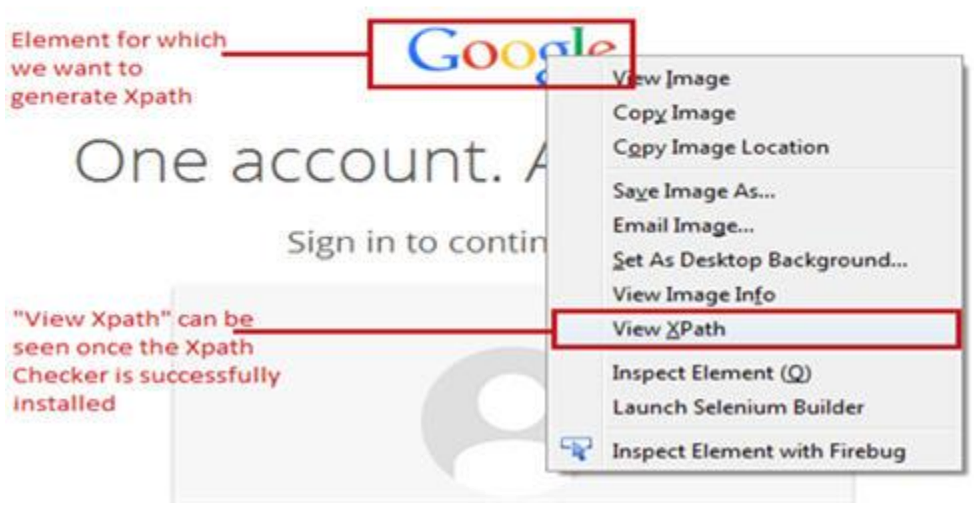
- The success rate of finding an element using Xpath is too high. Along with the previous statement, Xpath can find relatively all the elements within a web page. Thus, Xpaths can be used to locate elements having no id, class or name.
- Creating a valid Xpath is a tricky and complex process. There are plug-ins available to generate Xpath but most of the times, the generated Xpaths fails to identify the web element correctly.
- While creating xpath, user should be aware of the various nomenclatures and protocols.

<u>**Selenium Xpath Examples**</u>

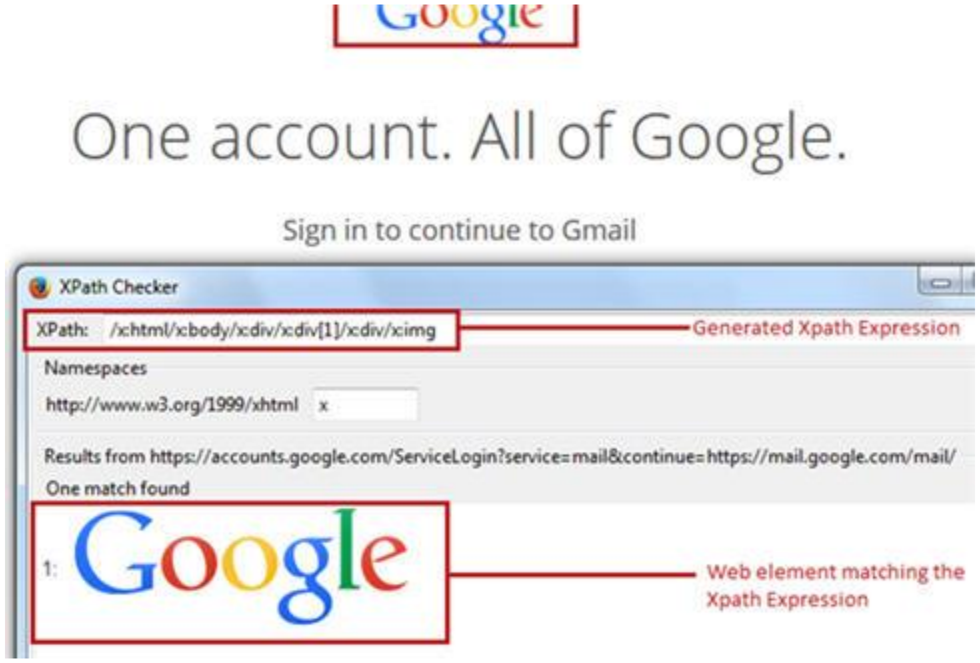
Xpath Checker

Creating Xpath becomes a little simpler by using Xpath Checker. Xpath Checker is a firefox add-on to automatically generate Xpath for a web element. The add-on can be downloaded and installed like any other plug-in. The plug-in can be downloaded from "https://addons.mozilla.org/en-US/firefox/addon/xpath-checker/".

As soon as the plug-in is installed, it can be seen in the context menu by right clicking any element whose xpath we want to generate.



Click on the “View Xpath” to see the Xpath expression of the element. An editor window would appear with the generated Xpath expression. Now user has the liberty to edit and modify the generated Xpath expression. The corresponding results would be updated cumulatively.


One account. All of Google.
Sign in to continue to Gmail
XPath Checker
XPath: /x:html/x:body/x:div/x:div[1]/x:div/x:img
Generated Xpath Expression
Namespaces
http://www.w3.org/1999/xhtml x
Results from https://accounts.google.com/ServiceLogin?service=mail&continue=https://mail.google.com/mail/
One match found
Web element matching the Xpath Expression

Note that the Xpath Checker is available for other browsers as well.

But re-iterating the fact, that most of the times, the generated Xpaths fails to identify the web element rightly. Thus, it is recommended to create our own Xpath following the pre defined rules and protocols.

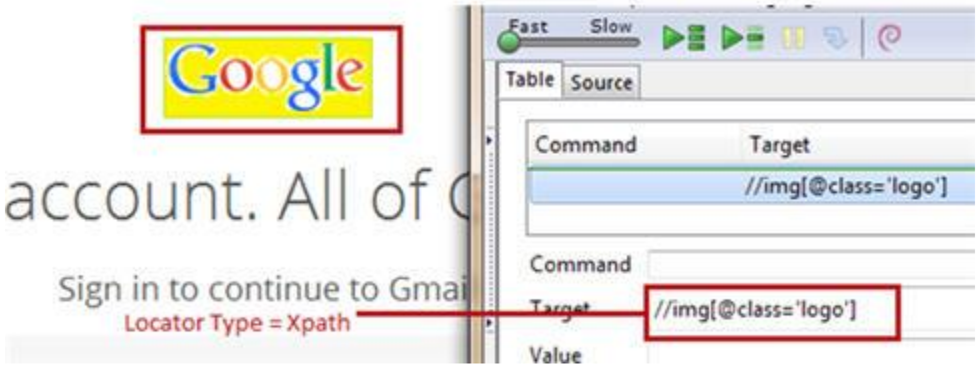
In this sample, we would access “Google” image present at the top of the login form at gmail.com.

<u>Creating a Xpath of a web element</u>

Step 1: Type “//img[@class=’logo’]” i.e. the locator value in the target box within the Selenium IDE.

<u>Syntax</u>: Xpath of the element

Step 2: Click on the Find Button. Notice that the image would be highlighted with yellow color with a florescent green border around the field.



Conclusion
Here are the cruxes of this article.

- Locators are the HTML properties of a web element which tells the Selenium about the web element on which it needs to perform actions.
- There is a wide range of web elements that a user may have to interact with on a regular basis. Some of them are: Text box, Button, Drop Down, Hyperlink, Check Box, and Radio Button.
- With the varied range of web elements comes a vast province of strategies/approaches to locate these web elements.
- Some of the extensively used locator types are: ID, ClassName, Link Text, Xpath, CSS Selectors and Name.

Note: Owing to the fact that creating CSS Selector and Xpath requires a lot of efforts and practice, thus the process is only exercised by more sophisticated and trained users.

In this tutorial we learned different types of locators including Selenium Xpath.
Next Tutorial #6: In continuation with this Selenium Locator types tutorial we will learn how to use **CSS Selector as a Locator.**
Any queries? Let us know in comments. We will try to resolve all.

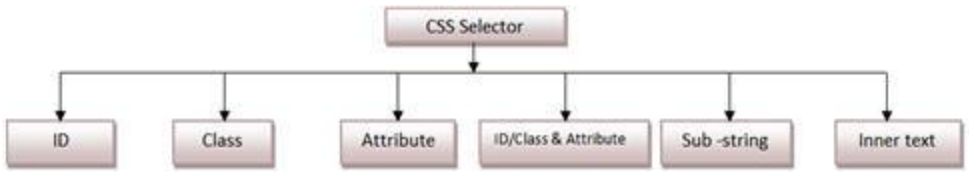
How to Use CSS Selector for Identifying Web Elements for Selenium Scripts – Selenium Tutorial #6

In our previous Selenium tutorialwe learned different types of locators. We also learned how to use: ID, ClassName, Name, Link Text, and Xpath locators for identifying web elements in a web page.
In continuation with that, today we will learn ***how to use CSS Selector as a Locator****. This is our 6th tutorial in our free Selenium Training series.*
Using CSS Selector as a Locator:
CSS Selector is combination of an element selector and a selector value which identifies the web element within a web page. The composite of element selector and selector value is known as Selector Pattern.

Selector Pattern is constructed using HTML tags, attributes and their values. The central theme behind the procedure to create CSS Selector and Xpath are very much similar underlying the only difference in their construction protocol.

Like Xpath, CSS selector can also locate web elements having no ID, class or Name.

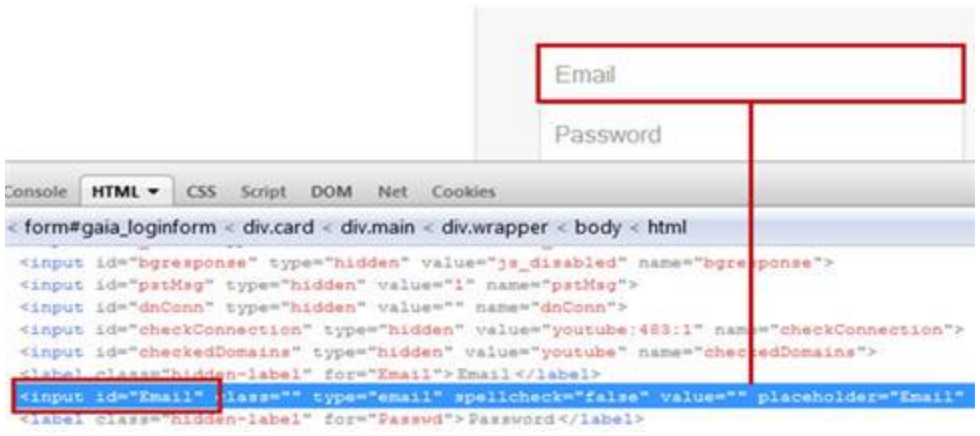
So now gearing ahead, let us discuss the primitive types of CSS Selectors:



CSS Selector: ID
In this sample, we would access "Email" text box present in the login form at Gmail.com.

The Email textbox has an ID attribute whose value is defined as "Email". Thus ID attribute and its value can be used to create CSS Selector to access the email textbox.

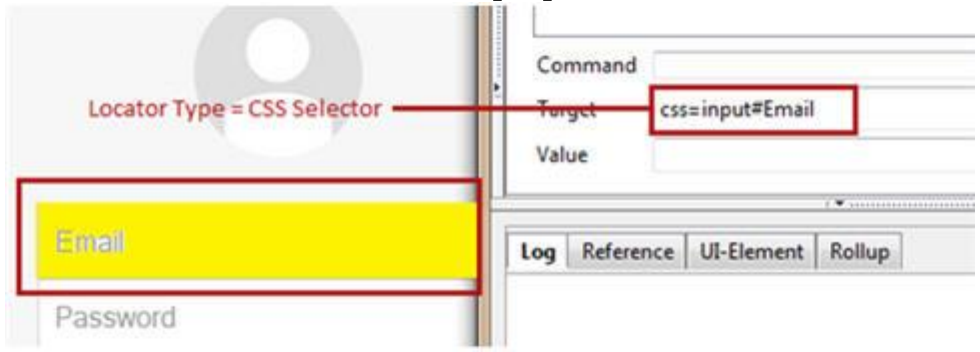
Creating CSS Selector for web element
Step 1: Locate / inspect the web element ("Email" textbox in our case) and notice that the html tag is "input" and value of ID attribute is "Email" and both of them collectively make a reference to the "Email Text box". Hence the above data would be used to create CSS Selector.


Email
Password
Console
HTML
CSS
Script
DOM
Net
Cookies
form#gaia_loginform < div.card < div.main < div.wrapper < body < html

```
<input id="bgresponse" type="hidden" value="js_disabled" name="bgresponse">
<input id="pstMsg" type="hidden" value="1" name="pstMsg">
<input id="dnConn" type="hidden" value="" name="dnConn">
<input id="checkConnection" type="hidden" value="youtube:483:1" name="checkConnection">
<input id="checkedDomains" type="hidden" value="youtube" name="checkedDomains">
<label class="hidden-label" for="Email">Email</label>
<input id="Email" class="" type="email" spellcheck="false" value="" placeholder="Email"
<label class="hidden-label" for="Passwd">Password</label>
```

Verify the locator value

Step 1: Type “css=input#Email” i.e. the locator value in the target box in the Selenium IDE and click on the Find button. Notice that the Email Text box would be highlighted.



Syntax

css=<HTML tag><#><Value of ID attribute>

- **HTML tag** – It is tag which is used to denote the web element which we want to access.
- **#** – The hash sign is used to symbolize ID attribute. It is mandatory to use hash sign if ID attribute is being used to create CSS Selector.
- **Value of ID attribute** – It is the value of an ID attribute which is being accessed.
- The value of ID is always preceded by a hash sign.

Note: Also applicable for other types of CSS Selectors

- While specifying CSS Selector in the target text box of Selenium IDE, always remember to prefix it with “css=”.
- The sequence of the above artifacts is inalterable.
- If two or more web elements have the same HTML tag and attribute value, the first element marked in the page source will be identified.

CSS Selector: Class

In this sample, we would access “Stay signed in” check box present below the login form at gmail.com.

The “Stay signed in” check box has a Class attribute whose value is defined as “remember”. Thus Class attribute and its value can be used to create CSS Selector to access the designated web element.

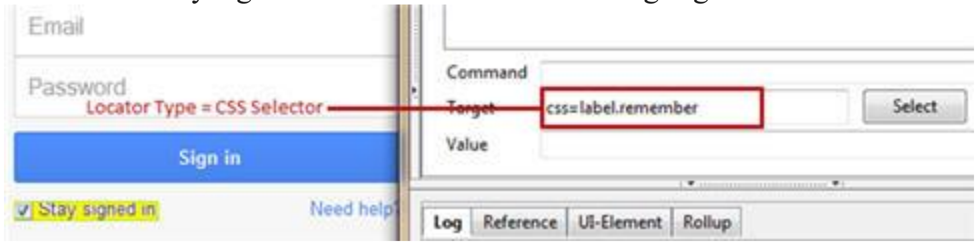
Locating an element using Class as a CSS Selector is very much similar to using ID, the lone difference lies in their syntax formation.

Creating CSS Selector for web element

Step 1: Locate / inspect the web element (“Stay signed in” check box in our case) and notice that the html tag is “label” and value of ID attribute is “remember” and both of them collectively make a reference to the “Stay signed in check box”.

Verify the locator value

Step 1: Type “css=label.remember” i.e. the locator value in the target box in the Selenium IDE and click on the Find Button. Notice that the “Stay signed in” check box would be highlighted.



Syntax

css=<HTML tag><.><Value of Class attribute>

- **.** – The dot sign is used to symbolize Class attribute. It is mandatory to use dot sign if Class attribute is being used to create CSS Selector.
- The value of Class is always preceded by a dot sign.

CSS Selector: Attribute

In this sample, we would access “Sign in” button present below the login form at gmail.com.

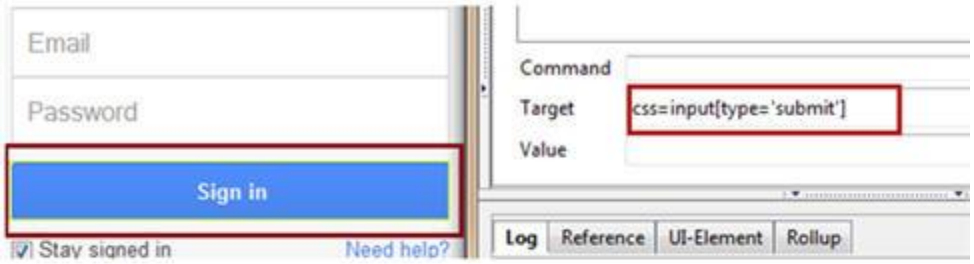
The “Sign in” button has a type attribute whose value is defined as “submit”. Thus type attribute and its value can be used to create CSS Selector to access the designated web element.

Creating CSS Selector for web element

Step 1: Locate / inspect the web element (“Sign in” button in our case) and notice that the html tag is “input”, attribute is type and value of type attribute is “submit” and all of them together make a reference to the “Sign in” button.

Verify the locator value

Step 1: Type “css=input[type=’submit’]” i.e. the locator value in the target box in the Selenium IDE and click on the Find Button. Notice that the “Sign in” button would be highlighted.



Syntax

css=<HTML tag><[attribute=Value of attribute]>

- **Attribute** – It is the attribute we want to use to create CSS Selector. It can value, type, name etc. It is recommended to choose an attribute whose value uniquely identifies the web element.
- **Value of attribute** – It is the value of an attribute which is being accessed.

CSS Selector: ID/Class and attribute

In this sample, we would access “Password” text box present in the login form at gmail.com.

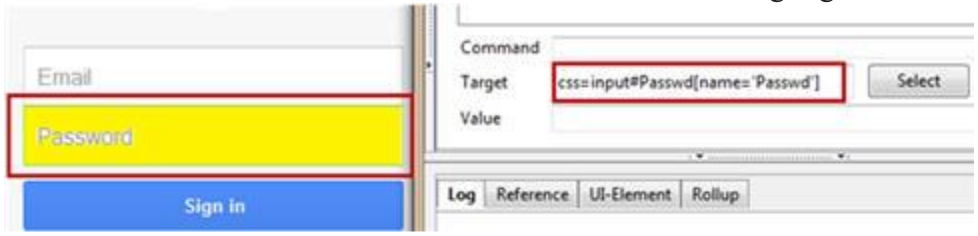
The “Password” text box has an ID attribute whose value is defined as “Passwd”, type attribute whose value is defined as “password”. Thus ID attribute, type attribute and their values can be used to create CSS Selector to access the designated web element.

Creating CSS Selector for web element

Step 1: Locate / inspect the web element (“Password” text box in our case) and notice that the html tag is “input”, attributes are ID and type and their corresponding values are ”Passwd” and “password” and all of them together make a reference to the “Password” textbox.

Verify the locator value

Step 1: Type “css=input#Passwd[name=’Passwd’]” i.e. the locator value in the target box in the Selenium IDE and click on the Find Button. Notice that the “Password” text box would be highlighted.



Syntax

css=<HTML tag><. Or #><value of Class or ID attribute><[attribute=Value of attribute]>

Two or more attributes can also be furnished in the syntax. For example, *“css=input#Passwd[type=’password’][name=’Passwd’]”.*

CSS Selector: Sub-string

CSS in Selenium allows matching a partial string and thus deriving a very interesting feature to create CSS Selectors using sub strings. There are three ways in which CSS Selectors can be created based on mechanism used to match the sub string.

Types of mechanisms

All the underneath mechanisms have symbolic significance.

- Match a prefix
- Match a suffix
- Match a sub string

Let us discuss them in detail.

Match a prefix

It is used to correspond to the string with the help of a matching prefix.

Syntax

css=<HTML tag><[attribute^=prefix of the string]>

- **^** – Symbolic notation to match a string using prefix.
- **Prefix** – It is the string based on which match operation is performed. The likely string is expected to start with the specified string.

For Example: Let us consider "Password textbox", so the corresponding CSS Selector would be:

css=input#Passwd[name^='Pass']

Match a suffix

It is used to correspond to the string with the help of a matching suffix.

Syntax

css=<HTML tag><[attribute$=suffix of the string]>

- **#** – Symbolic notation to match a string using suffix.
- **Suffix** – It is the string based on which match operation is performed. The likely string is expected to ends with the specified string.

For Example: Lets again consider "Password textbox", so the corresponding CSS Selector would be:

css=input#Passwd[name$='wd']

Match a sub string

It is used to correspond to the string with the help of a matching sub string.

Syntax

css=<HTML tag><[attribute*=sub string]>

- ***** – Symbolic notation to match a string using sub string.
- **Sub string** – It is the string based on which match operation is performed. The likely string is expected to have the specified string pattern.

For Example: Lets again consider "Password textbox", so the corresponding CSS Selector would be:
css=input#Passwd[name$='wd']

CSS Selector: Inner text

Inner text helps us identify and create CSS Selector using a string pattern that the HTML Tag manifests on the web page.

Consider, "Need help?" hyperlink present below the login form at gmail.com.

The anchor tag representing the hyperlink has a text enclosed within. Thus this text can be used to create CSS Selector to access the designated web element.

Syntax

css=<HTML tag><:><contains><(text)>

- **:** – The dot sign is used to symbolize contains method
- Contains – It is the value of a Class attribute which is being accessed.
- Text – The text that is displayed anywhere on the web page irrespective of its location.

This is one of the most frequently used strategies to locate web element because of its simplified syntax.

Owing to the fact that creating CSS Selector and Xpath requires a lot of efforts and practice, thus the process is only exercised by more sophisticated and trained users.

Next Tutorial #7: Proceeding ahead with our next tutorial, we would take the opportunity to introduce you with an extension of locating strategies. Thus, in the next tutorial, we would study the **mechanism to locate web elements on Google Chrome and Internet Explorer.**

We are covering Selenium Locators in more details as it is important part of Selenium Script creation.

How to Locate Elements in Chrome and IE Browsers for Building Selenium Scripts – Selenium Tutorial #7

This is tutorial #7 in our Selenium Online Training Series. If you want to check all Selenium tutorials in this series please check ***this page****.*

In the previous tutorial, we tried to shed light on various types of locators in Selenium and their locating mechanisms to build test scripts. The tutorial was primary consist of the brief introduction of different locator types like ID, Classes, Xpaths, Link texts, CSS Selectors etc. and their identification.

Proceeding ahead with our next tutorial, we would take the opportunity to introduce you with an extension of locating strategies. Thus, in the next tutorial, **we would study the mechanism to locate web elements on Google Chrome and Internet Explorer.**

As we all are well aware of the fact that there is rapid growth in the internet user base, thus stakeholders and programmers are building web applications which are likely to work on most of the browsers. Thus, imagine a situation where your web application doesn't support Firefox but works well for Chrome and Internet Explorer. Now how are you going to automate such an application using Selenium? Or to be specific how are you going to locate web elements in Chrome and Internet Explorer. Thus the answer lies ahead in this tutorial.

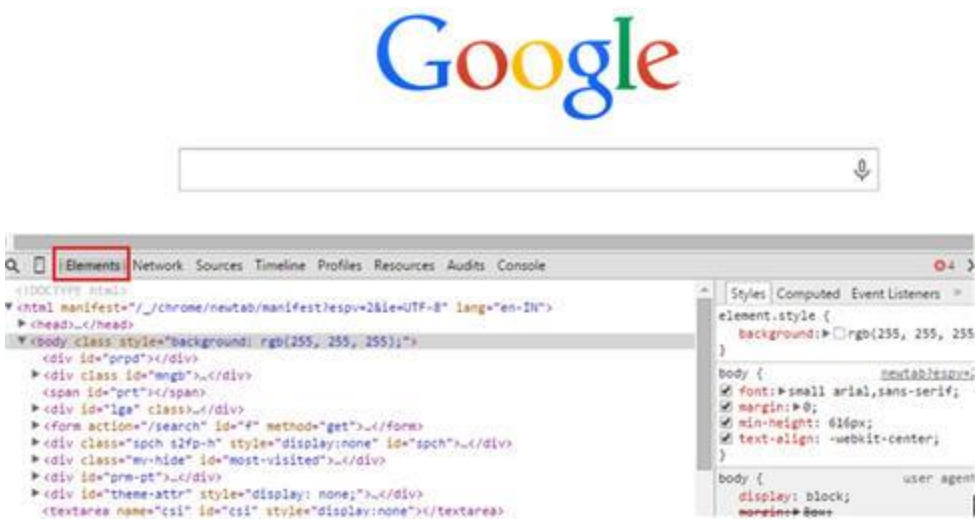
Locating Web Elements in Google Chrome

Let us begin with understanding the locating strategies in Google Chrome.

Like firebug in Firefox, Google Chrome has its own developer tool that can be used to identify and locate web elements on the web page. Unlike firebug, user is not required to download or install any separate plugin; the developer tool comes readily bundled with Google Chrome.

Follow the below steps to locate web elements using Chrome's Developer tool:

Step #1: The primary step is to launch the Google Chrome's Developer tool. Press F12 to launch the tool. The user would be able to see something like the below screen.



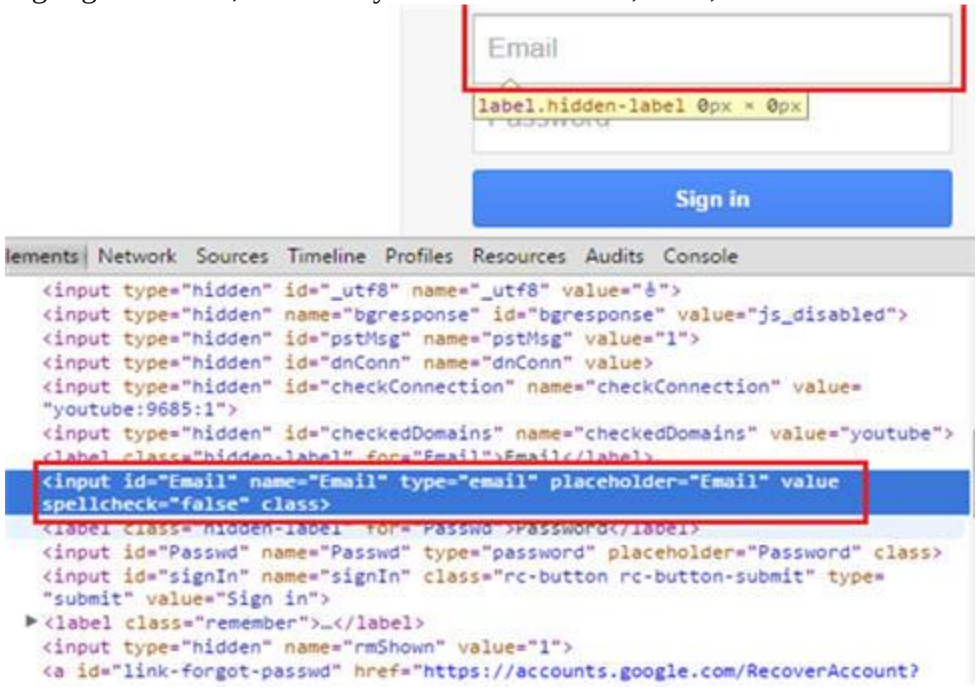
```
<!DOCTYPE html>
▼<html manifest="/_/chrome/newtab/manifest?espv=2&ie=UTF-8" lang="en-IN">
  ▶<head>…</head>
  ▼<body class style="background: rgb(255, 255, 255);">
     <div id="prpd"></div>
    ▶<div class id="mngb">…</div>
     <span id="prt"></span>
    ▶<div id="lga" class>…</div>
    ▶<form action="/search" id="f" method="get">…</form>
    ▶<div class="spch s2fp-h" style="display:none" id="spch">…</div>
    ▶<div class="mv-hide" id="most-visited">…</div>
    ▶<div id="prm-pt">…</div>
    ▶<div id="theme-attr" style="display: none;">…</div>
     <textarea name="csi" id="csi" style="display:none"></textarea>
```

```
element.style {
  background:▶□rgb(255, 255, 255
}
body {
✓ font:▶small arial,sans-serif;
✓ margin:▶0;
✓ min-height: 616px;
✓ text-align: -webkit-center;
}
body {                                  user agent
  display: block;
  ~~margin:▶8px;~~
```

Take a note that “Element” tab is highlighted in the above screenshot. Thus, element tab is the one which displays all the HTML properties belonging to the current web page. Navigate to the “Element” tab if it is not opened by default on the launch.

You can also launch developer tool by right clicking anywhere within the web page and by selecting “Inspect element” which is very similar to that of firebug’s inspection.

Step #2: The next step is to locate the desired object within the web page. One way to do the same is to right click on the desired web element and inspect. The HTML property belonging to that web element would be highlighted in the developer tool. Another way is to hover through the HTML properties and the matching web element would be highlighted. Thus, in this way user can locate ids, class, links etc.

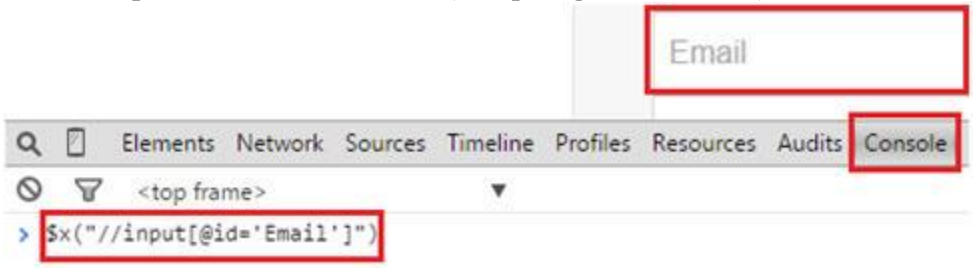


```
<input type="hidden" id="_utf8" name="_utf8" value="☃">
<input type="hidden" name="bgresponse" id="bgresponse" value="js_disabled">
<input type="hidden" id="pstMsg" name="pstMsg" value="1">
<input type="hidden" id="dnConn" name="dnConn" value>
<input type="hidden" id="checkConnection" name="checkConnection" value=
"youtube:9685:1">
<input type="hidden" id="checkedDomains" name="checkedDomains" value="youtube">
<label class="hidden-label" for="Email">Email</label>
<input id="Email" name="Email" type="email" placeholder="Email" value
spellcheck="false" class>
<label class="hidden-label" for="Passwd">Password</label>
<input id="Passwd" name="Passwd" type="password" placeholder="Password" class>
<input id="signIn" name="signIn" class="rc-button rc-button-submit" type=
"submit" value="Sign in">
▶<label class="remember">…</label>
<input type="hidden" name="rmShown" value="1">
<a id="link-forgot-passwd" href="https://accounts.google.com/RecoverAccount?
```

Creating an Xpath in Developer Tool

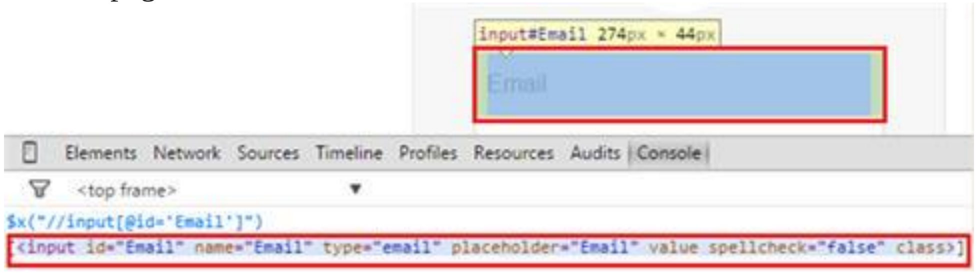
We have already discussed about Xpaths in the last tutorial. We also discussed about its creation strategy. Here we would base our discussion to check the validity of the created Xpath in Chrome's Developer tool.

Step #1: For creating Xpath in Developer tool, open the console tab.
Step #2: Type the created Xpath and enclose it in $x("//input[@id='Email']")



```
> $x("//input[@id='Email']")
```

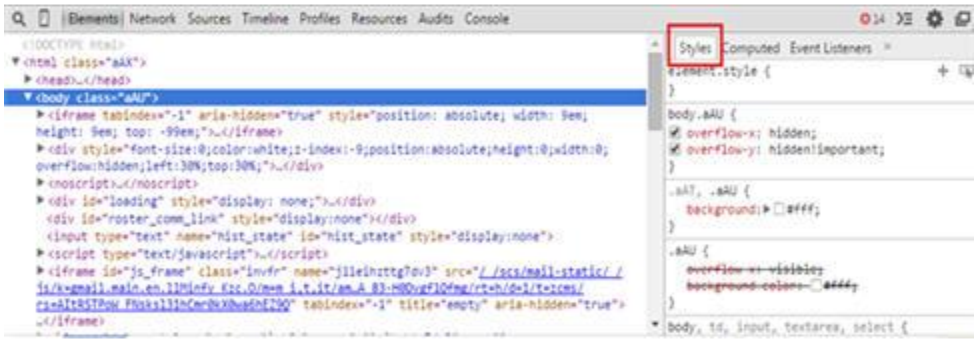
Step #3: Press the enter key to see all the matching HTML elements with the specified Xpath. In our case, there is only one matching HTML element. Hover on that HTML element and the corresponding web element would be highlighted on the web page.



```
$x("//input[@id='Email']")
[<input id="Email" name="Email" type="email" placeholder="Email" value spellcheck="false" class>]
```

In this way, all the Xpaths can be created and checked for their validity within the console.

Information related to CSS corresponding to the web element can be found within the Chrome's Developer tool. Refer the screenshot below:

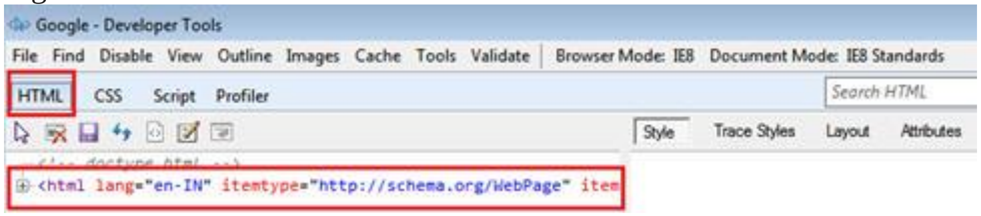


```
<!DOCTYPE html>
▼<html class="aAX">
  ▶<head>…</head>
  ▼<body class="aAU">
    ▶<iframe tabindex="-1" aria-hidden="true" style="position: absolute; width: 9em;
    height: 9em; top: -99em;">…</iframe>
    ▶<div style="font-size:0;color:white;z-index:-9;position:absolute;height:0;width:0;
    overflow:hidden;left:30%;top:30%;">…</div>
    ▶<noscript>…</noscript>
    ▶<div id="loading" style="display: none;">…</div>
     <div id="roster_comm_link" style="display:none"></div>
     <input type="text" name="hist_state" id="hist_state" style="display:none">
    ▶<script type="text/javascript">…</script>
    ▶<iframe id="js_frame" class="invfr" name="j1leihzttg7ov3" src="/_/scs/mail-static/_/
    js/k=gmail.main.en.1lMinfy_Kzc.O/m=m_i,t,it/am=A_83-H0OvgF1Qfmg/rt=h/d=1/t=zcms/
    rs=AItRSTPoW_FNsks1l1hCmr0kX0wa6hEZ9Q" tabindex="-1" title="empty" aria-hidden="true">
    …</iframe>
```

```
element.style {
}
body.aAU {
  overflow-x: hidden;
  overflow-y: hidden!important;
}
.aAT, .aAU {
  background: ▶ #fff;
}
.aAU {
  ~~overflow-x: visible;~~
  ~~background-color: #fff;~~
}
body, td, input, textarea, select {
```

Locating Web Elements in Internet Explorer
Like Google Chrome, Internet Explorer also has its own Developer Toolthat can be used to identify web elements based on their properties within the web page. User is not required to download or install any separate plugin, the developer tool comes readily bundled with Internet Explorer.
Follow the below steps to locate web elements using IE Developer tool:

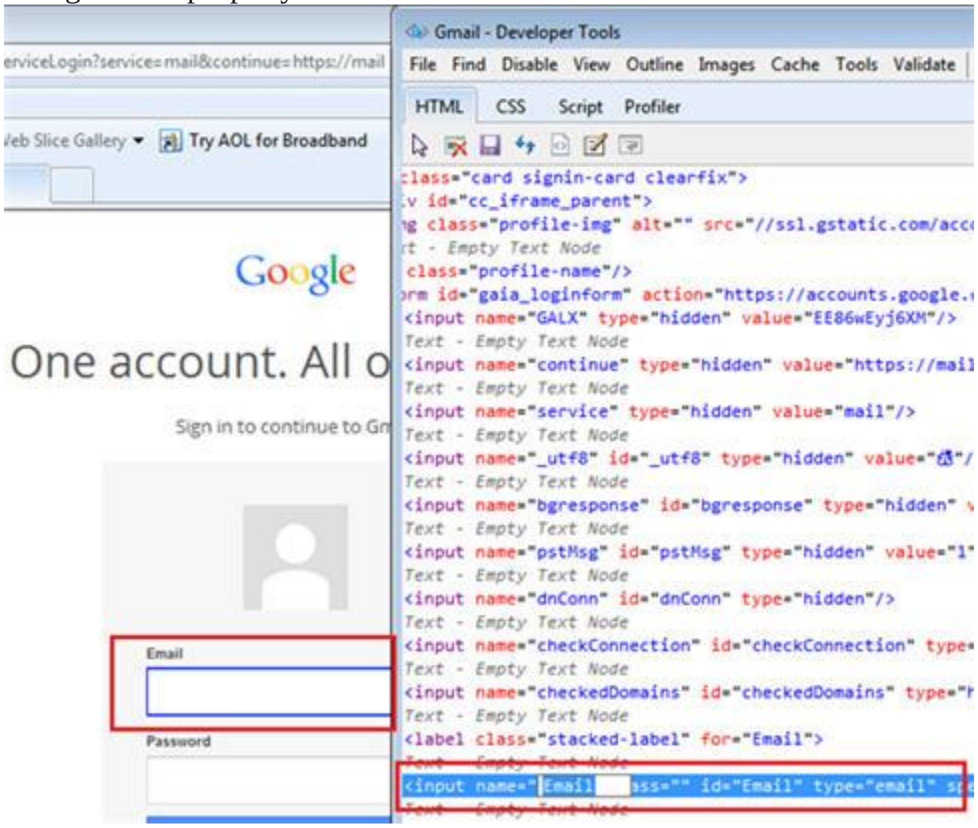
Step #1: The primary step is to launch the IE Developer tool. Press F12 to launch the tool. The user would be able to see something like the below screen.


Google - Developer Tools
File Find Disable View Outline Images Cache Tools Validate Browser Mode: IE8 Document Mode: IE8 Standards
HTML CSS Script Profiler
Search HTML
Style Trace Styles Layout Attributes

```
<!-- doctype html -->
<html lang="en-IN" itemtype="http://schema.org/WebPage" item
```

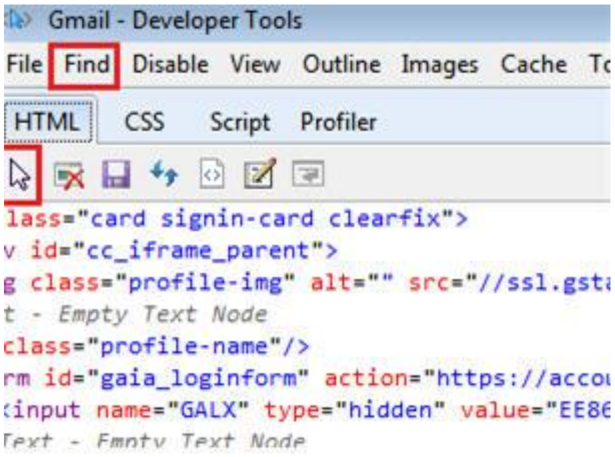
Take a note that “HTML” tab is highlighted in the above screenshot. Thus, HTML tab is the one which displays all the HTML properties belonging to the current web page. Expand the HTML tab to view the properties of all the web elements belonging to the current web page.

Step #2: The next step is to locate the desired object within the web page. One way to this is to select the HTML element and the matching web element would be highlighted. Thus, in this way user can locate ids, class, links etc. Check out in the below screenshot in which Email Textbox would be highlighted as soon as we select the corresponding HTML property.


erviceLogin?service=mail&continue=https://mail
Veb Slice Gallery ▾
Try AOL for Broadband
Google
One account. All o
Sign in to continue to Gr
Email
Password
Gmail - Developer Tools
File Find Disable View Outline Images Cache Tools Validate
HTML CSS Script Profiler

```
class="card signin-card clearfix">
v id="cc_iframe_parent">
g class="profile-img" alt="" src="//ssl.gstatic.com/acc
t - Empty Text Node
class="profile-name"/>
rm id="gaia_loginform" action="https://accounts.google.
<input name="GALX" type="hidden" value="EE86wEyj6XM"/>
Text - Empty Text Node
<input name="continue" type="hidden" value="https://mail
Text - Empty Text Node
<input name="service" type="hidden" value="mail"/>
Text - Empty Text Node
<input name="_utf8" id="_utf8" type="hidden" value="☃"/
Text - Empty Text Node
<input name="bgresponse" id="bgresponse" type="hidden"
Text - Empty Text Node
<input name="pstMsg" id="pstMsg" type="hidden" value="1"
Text - Empty Text Node
<input name="dnConn" id="dnConn" type="hidden"/>
Text - Empty Text Node
<input name="checkConnection" id="checkConnection" type=
Text - Empty Text Node
<input name="checkedDomains" id="checkedDomains" type="h
Text - Empty Text Node
<label class="stacked-label" for="Email">
Text - Empty Text Node
<input name="Email ass="" id="Email" type="email" s
Text - Empty Text Node
```

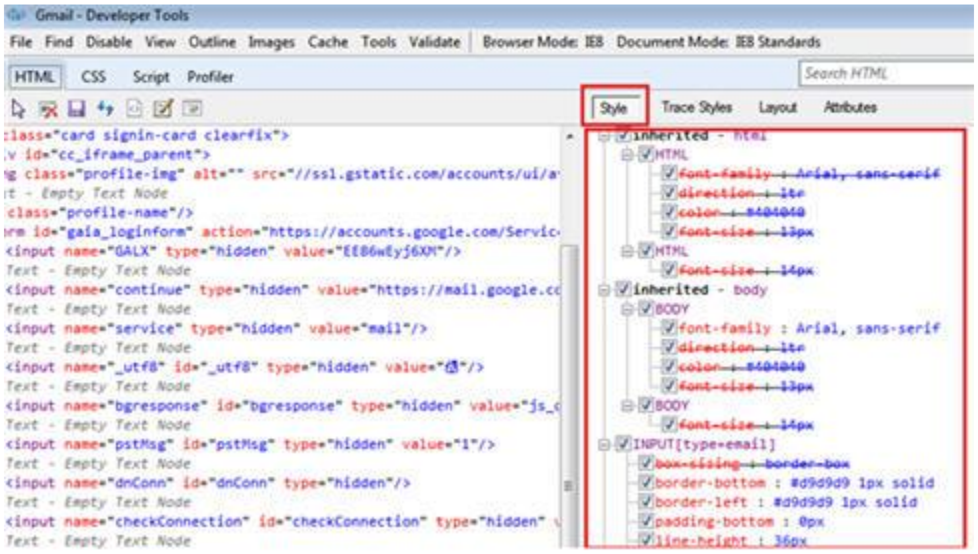
Another way to locate the web element is to click on the “Find” button present in the top menu and by clicking on the desired web element within the web page. As a result, the corresponding HTML properties would be highlighted.



```
lass="card signin-card clearfix">
v id="cc_iframe_parent">
g class="profile-img" alt="" src="//ssl.gsta
t - Empty Text Node
class="profile-name"/>
rm id="gaia_loginform" action="https://accou
<input name="GALX" type="hidden" value="EE86
Text - Empty Text Node
```

Thus, by using the developer tool, user can find ids, classes, tag names and can create Xpaths to locate web elements.

Like Chrome’s Developer tool, IE developer tool has a separate section that displays CSS related information. Checkout the below screenshot.


Gmail - Developer Tools
File Find Disable View Outline Images Cache Tools Validate Browser Mode: IE8 Document Mode: IE8 Standards
HTML CSS Script Profiler
Search HTML
Style Trace Styles Layout Attributes

```
class="card signin-card clearfix">
v id="cc_iframe_parent">
g class="profile-img" alt="" src="//ssl.gstatic.com/accounts/ui/a
t - Empty Text Node
class="profile-name"/>
rm id="gaia_loginform" action="https://accounts.google.com/Servic
<input name="GALX" type="hidden" value="EE86wEyj6XM"/>
Text - Empty Text Node
<input name="continue" type="hidden" value="https://mail.google.cc
Text - Empty Text Node
<input name="service" type="hidden" value="mail"/>
Text - Empty Text Node
<input name="_utf8" id="_utf8" type="hidden" value="☃"/>
Text - Empty Text Node
<input name="bgresponse" id="bgresponse" type="hidden" value="js_c
Text - Empty Text Node
<input name="pstMsg" id="pstMsg" type="hidden" value="1"/>
Text - Empty Text Node
<input name="dnConn" id="dnConn" type="hidden"/>
Text - Empty Text Node
<input name="checkConnection" id="checkConnection" type="hidden"
Text - Empty Text Node
```

```
inherited - html
  HTML
    ~~font-family : Arial, sans-serif~~
    ~~direction : ltr~~
    ~~color : #404040~~
    ~~font-size : 13px~~
  HTML
    ~~font-size : 14px~~
inherited - body
  BODY
    font-family : Arial, sans-serif
    ~~direction : ltr~~
    ~~color : #404040~~
    ~~font-size : 13px~~
  BODY
    ~~font-size : 14px~~
INPUT[type=email]
  ~~box-sizing : border-box~~
  border-bottom : #d9d9d9 1px solid
  border-left : #d9d9d9 1px solid
  padding-bottom : 0px
  line-height : 36px
```

Conclusion

In this tutorial, we shed light on the basic element locating strategies using Developer’s tool for Google Chrome and Internet Explorer.

Next Tutorial #8: Proceeding ahead with our next tutorial, we would take the pleasure to introduce you with a more advanced tool named as WebDriver. WebDriver is one of the most compelling automation testing tools. So our next tutorial onwards, we would route and base our discussions around WebDriver and all its nitty gritty.

Selenium WebDriver:

Selenium WebDriver

- Tutorial #8 – Selenium WebDriver Introduction (Must Read)
- Tutorial #9 – Selenium WebDriver Installation with eclipse
- Tutorial #10 – My first Selenium WebDriver script (Must Read)
- Tutorial #11 – Introduction to JUnit
- Tutorial #12 – Introduction to TestNG (Must Read)
- Tutorial #13 – Handling Drop-downs
- Tutorial #14 – Looping and Conditional commands
- Tutorial #15 – Explicit and Implicit Waits
- Tutorial #16 – Handling Alerts/popups
- Tutorial #17 – Commonly used commands
- Tutorial #18 – Handling Web Tables, Frames, Dynamic Elements
- Tutorial #19 – Exception Handling

Introduction to Selenium WebDriver – Selenium Tutorial #8

Introduction to Selenium WebDriver:

Earlier in this series we published tutorials which focused more on Selenium IDE and its various aspects. We introduced the tool and discussed about its features. We also constructed a few scripts using Selenium IDE and Firebug. From there we moved on to different types of web elements available and their locating strategies.

Now that we are well versed with Selenium IDE, **let us move our learning curve towards creating more advanced automation scripts using *Selenium WebDriver*.** WebDriver is one of the most compelling automation testing tools. Let us discuss it in detail.

Introduction

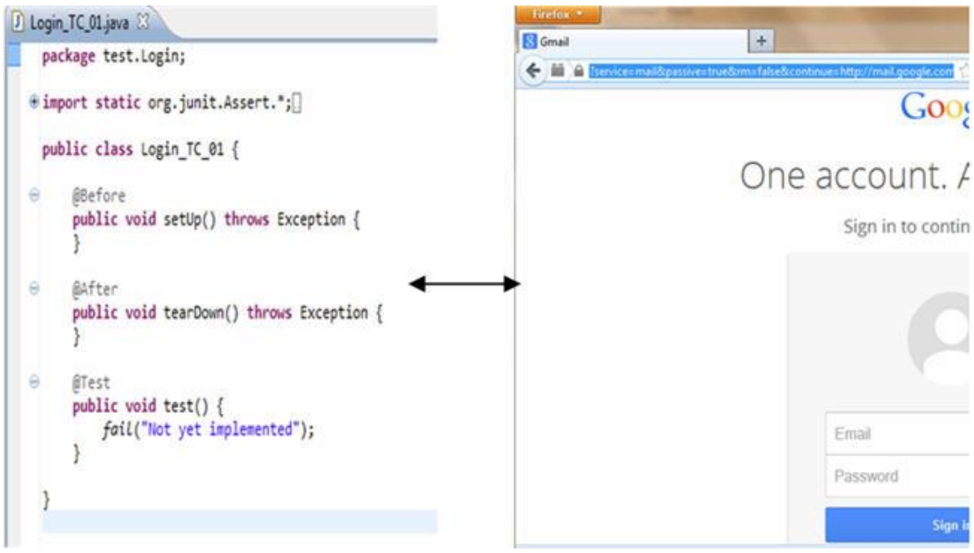
WebDriver is one of the most powerful and popular tools of Selenium toolkit. WebDriver comes as an extended version to Selenium RC with superfluous advantages and addresses many of its limitations. WebDriver extends its support to many latest browsers and platforms unlike Selenium IDE. WebDriver also doesn't require Selenium server to be started prior to execution of the test scripts unlike Selenium RC.

Selenium RC in aggregation with WebDriver API is known as Selenium 2.0. Selenium was so developed in order to support dynamic web pages and Ajax calls. It also supports various drivers to exercise web based mobile testing.

Architecture

WebDriver is a web-based testing tool with a subtle difference with Selenium RC. Since, the tool was built on the fundamental where an isolated client was created for each of the web browser; no JavaScript Heavy lifting was required as we discussed in our very first tutorial.

WebDriver makes direct calls to the Web browser and the entire test script is executed in this fashion. WebDriver uses the browsers support and capabilities to automation.



```
package test.Login;

import static org.junit.Assert.*;

public class Login_TC_01 {

    @Before
    public void setUp() throws Exception {
    }

    @After
    public void tearDown() throws Exception {
    }

    @Test
    public void test() {
        fail("Not yet implemented");
    }

}
```

Unlike Selenium RC, Selenium WebDriver doesn't essentially require Selenium Server to be started before launching the test script execution. User can leverage the benefit and may or may not require Selenium Server if he/she desires to perform the test execution on the same machine where the browser is residing.

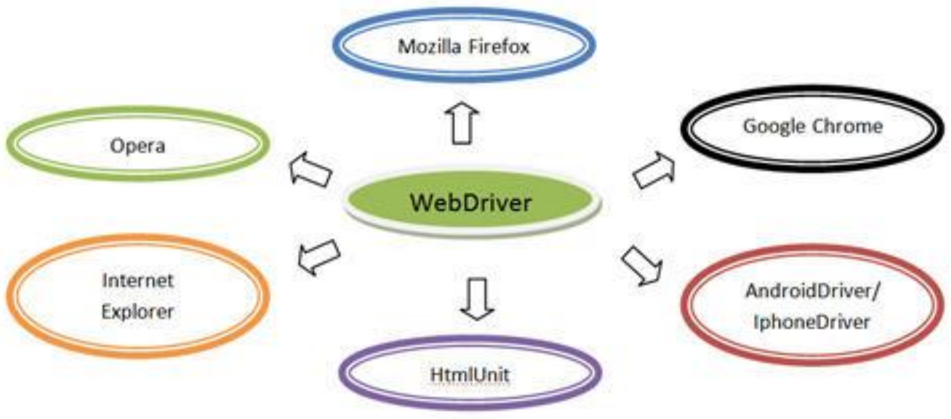
Exceptional Cases when Selenium Server is required with WebDriver:

- When the user wish to execute test scripts on the remote machine.
- When the user wish to execute test scripts on HtmlUnit Driver.
- When the user wish to execute test scripts on multiple platforms.

WebDriver is a purely object oriented framework that works on OS layer. It utilizes the browser's native compatibility to automation without using any peripheral entity. With the increasing demand it has gained a large popularity, user base and has become by far one of the most extensively used open source automation testing tool.

Features of Selenium WebDriver

<u>Browser Compatibility</u>



Browser Compatibility - © www.SoftwareTestingHelp.com

WebDriver supports diverse range of web browsers and their versions. It supports all the conventional browsers in addition to some unique and rare browsers like HtmlUnit browser unlike Selenium RC and Selenium IDE.

HtmlUnit Browser executes the test scripts analogous to other browsers except the fact that it runs in the headless mode i.e. GUI-less mode and the user won’t be able to view the test script execution. Said that the test script execution transpires in headless mode, thus the execution speed takes a roll and quickens the execution.

WebDriver also supports web based mobile testing. Thus it provides AndroidDriver and IphoneDriver to back web based mobile testing.

Note: *WebDriver doesn’t readily support new browsers.*

Language Support

Earlier in the sessions we learned to create scripts using record and playback functionality. We also saw how to create them manually using Selenese commands. While creating such test scripts, we come across various constraints.

Some of the limitations imposed by Selenium IDE are:

- Doesn’t support iterations and conditional statements
- Doesn’t support loops
- Doesn’t support error handling
- Doesn’t support test script dependency

The above impediments can be troubleshot programmatically. WebDriver facilitates the user to choose within the different programming languages and build their test script in the designated language.

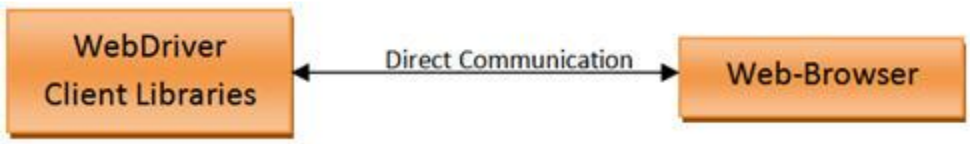
Selenium WebDriver supported programming languages are:

1. Java
2. C#
3. PHP
4. Pearl
5. Ruby
6. Python

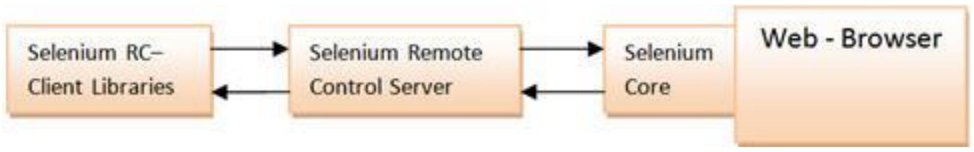
Thus the user can pick any one of the programming language (provided the language is supported by WebDriver) based on his/her competency and can start building test scripts.

Speed

When compared to other tools of Selenium suite, WebDriver turns out to be the fastest tool amongst all. The communication is not channelized via any external intervention; rather the tool directly communicates with the browser same as that of any user. Thus, WebDriver takes advantage of the browser’s native compatibility towards automation.

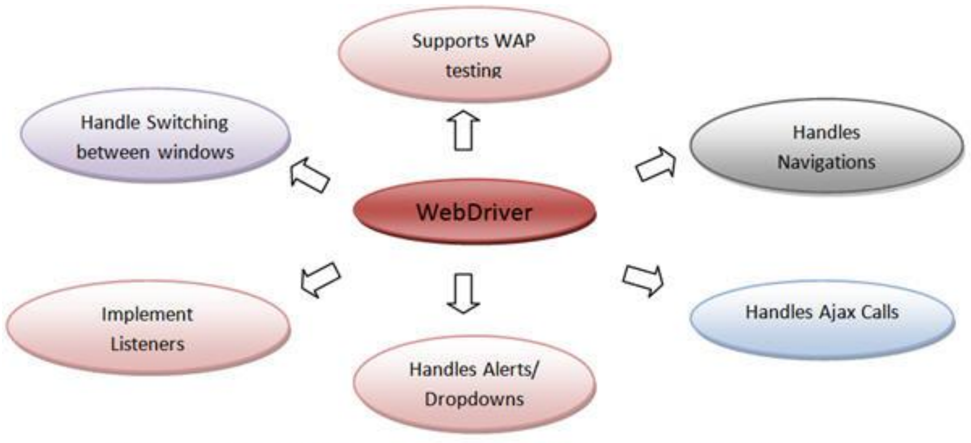


Other tools from Selenium suite like Selenium RC don't communicate directly with the web browser. Client libraries (test scripts written in any programming language) communicate with Selenium Remote Control Server and Remote Control communicates with a Selenium Core (JavaScript Program) which in turn communicates with the web browser. Hence, this sort of twisted communication results as a hindrance on execution speed.



Drivers, Methods and Classes

WebDriver offers a wide range of solutions to some potential challenges in Automation Testing. It helps us to deal with complex types of web elements like checkboxes, dropdowns, and alerts with the help of dynamic finders.



Drivers, Methods and Classes - © www.SoftwareTestingHelp.com

With the advent of mobile era, WebDriver API has also matured and introduced some of the key technologies to enter this horizon. WebDriver enables user to perform web based mobile testing. It provides two of the essentials drivers to perform web based mobile testing.

- AndriodDriver
- IphoneDriver

Moreover, WebDriver API is fairly simple and easy. It doesn't include repetitious commands. On the contrary, Selenium RC embodies many of the tautological commands.

Conclusion

In this tutorial, we tried to make you acquainted with Selenium WebDriver by outlining its architecture, features and limitations.

Here are the cruxes of this article.

- Selenium suite is comprised of 4 basic components; Selenium IDE, Selenium RC, WebDriver, Selenium Grid.
- WebDriver allows user to perform web based automation testing. WebDriver is a different tool altogether that has various advantages over Selenium RC.
- WebDriver supports a wide range of web browsers, programming languages and test environments.
- WebDriver directly communicates with the web browser and uses its native compatibility to automate.
- WebDriver's support doesn't only limits in the periphery of traditional user actions. Instead it supports efficient handling mechanisms for complex user actions like dealing with dropdowns, Ajax calls, switching between windows, navigation, handling alerts etc.
- WebDriver enables user to perform web based mobile testing. To support the same, WebDriver introduces AndroidDriver and IphoneDriver.
- WebDriver is faster than other tools of Selenium Suite because it makes direct calls to browser without any external intervention.

Next Tutorial #9: In the next tutorial, we would be discussing about the **installation procedure to get started with WebDriver** initiating from the scratch. We would also be discussing about the **diverse range of drivers provided by WebDriver**, each catering to different needs.

Till the time our next tutorial is under construction, the readers can visit the Selenium's official website. A detailed documentation with reference to Selenium WebDriver is implemented at its official website.

About the author: *Shruti Shrivastava is currently working as a Senior Test Engineer with 4+ years of automation testing experience. She is an ISTQB certified professional and also an active blogger, always interested in solving testing related problems.*

We also have two more authors on this Selenium tutorial's series to make it complete, useful and relevant.

If you have any specific requests/queries about this or any other tutorial in this Selenium online training series, let us know in comments.

WebDriver Entire Setup and Installation with Eclipse – Selenium Tutorial #9

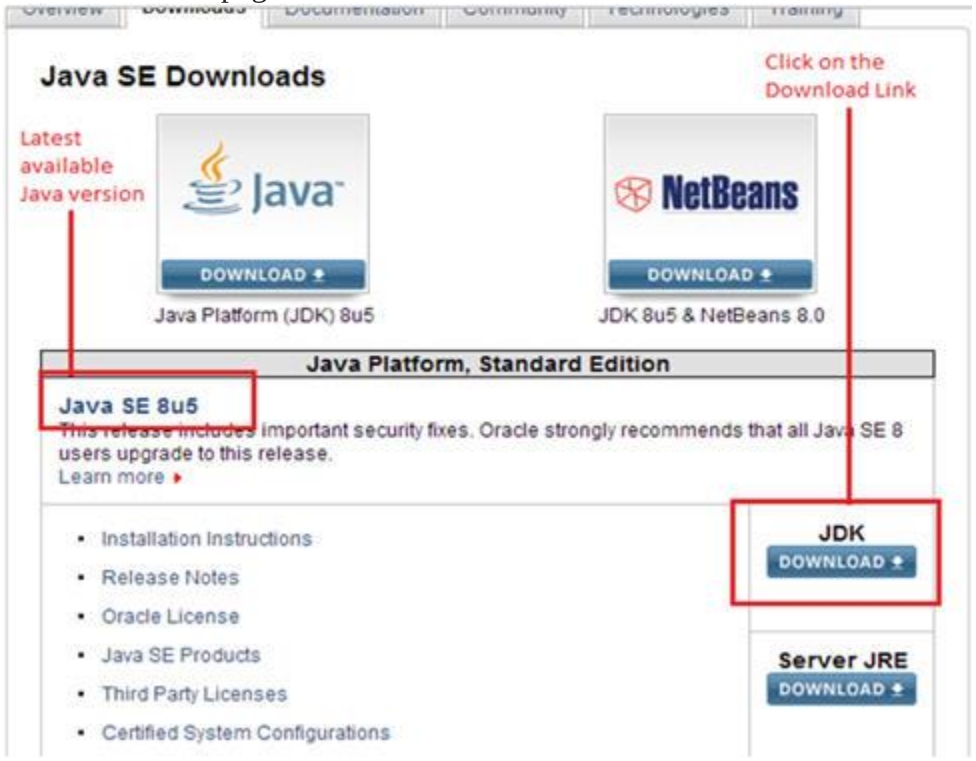
In the previous tutorial, we introduced the basic architecture and features of WebDriver. This is 9th tutorial in Selenium Tutorial Training Series.

In this tutorial, we would be discussing about the **installation procedure to get started with WebDriver initiating from the scratch**. We would also be discussing about the diverse range of drivers provided by WebDriver, each catering to different testing and environmental needs.

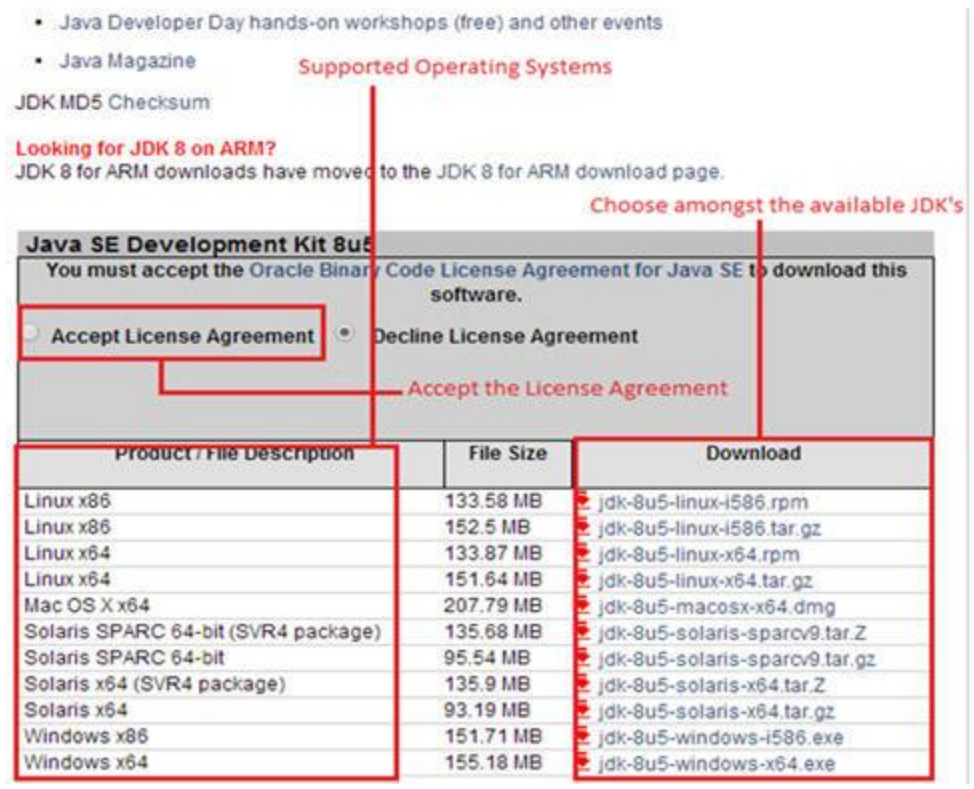
To be able to use WebDriver for scripting, there are some pre-requisites that need to be in place like the basic environment setup. In this series, we would be using Java as a programming language within our sample examples. Thus let us kick start with the Java installation.

Java Installation

Step 1: Go to Oracle official site – "JAVA download", download Java Platform, Standard Edition. All the recent releases are available on the page.


Java SE Downloads
Click on the Download Link
Latest available Java version
Java
DOWNLOAD
Java Platform (JDK) 8u5
NetBeans
DOWNLOAD
JDK 8u5 & NetBeans 8.0
Java Platform, Standard Edition
Java SE 8u5
This release includes important security fixes. Oracle strongly recommends that all Java SE 8 users upgrade to this release.
Learn more
Installation Instructions
Release Notes
Oracle License
Java SE Products
Third Party Licenses
Certified System Configurations
JDK
DOWNLOAD
Server JRE
DOWNLOAD

Step 2: As soon as you click on the Download button, following screen would appear. Accept the License agreement for Java installation and choose amongst the various catalogued Java Development Kit's. Select the one that best suits your system configuration.


Supported Operating Systems
Choose amongst the available JDK's
Accept the License Agreement

- Java Developer Day hands-on workshops (free) and other events
- Java Magazine

JDK MD5 Checksum

Looking for JDK 8 on ARM?
JDK 8 for ARM downloads have moved to the JDK 8 for ARM download page.

Java SE Development Kit 8u5
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Accept License Agreement   Decline License Agreement

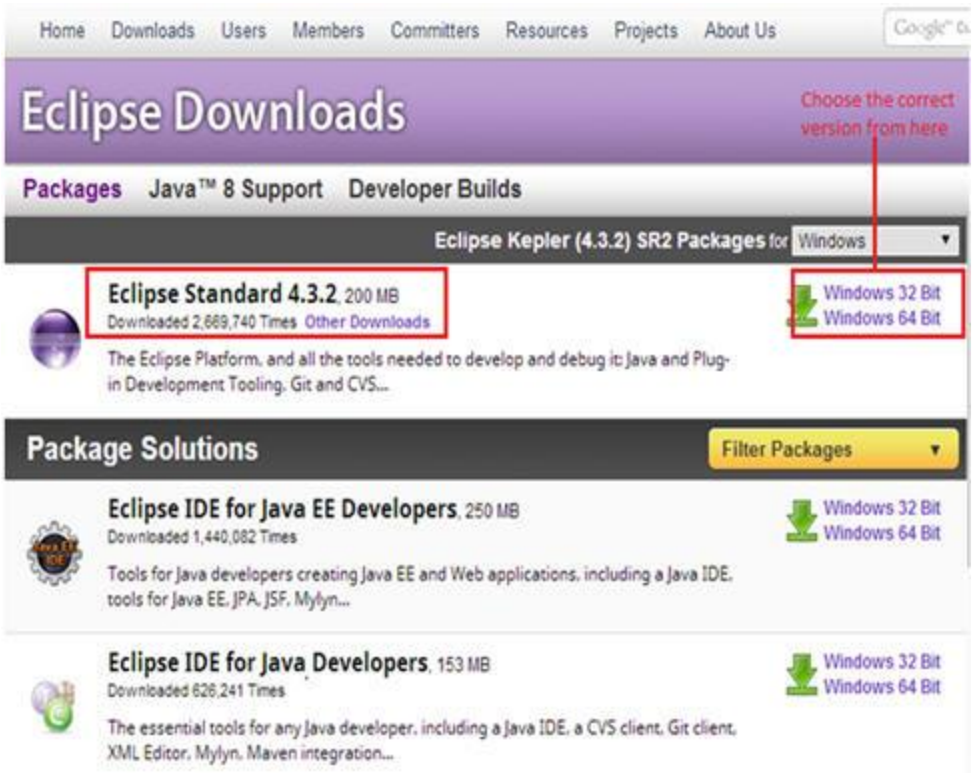
Product / File Description	File Size	Download
Linux x86	133.58 MB	jdk-8u5-linux-i586.rpm
Linux x86	152.5 MB	jdk-8u5-linux-i586.tar.gz
Linux x64	133.87 MB	jdk-8u5-linux-x64.rpm
Linux x64	151.64 MB	jdk-8u5-linux-x64.tar.gz
Mac OS X x64	207.79 MB	jdk-8u5-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	135.68 MB	jdk-8u5-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	95.54 MB	jdk-8u5-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	135.9 MB	jdk-8u5-solaris-x64.tar.Z
Solaris x64	93.19 MB	jdk-8u5-solaris-x64.tar.gz
Windows x86	151.71 MB	jdk-8u5-windows-i586.exe
Windows x64	155.18 MB	jdk-8u5-windows-x64.exe

Remember to download JDK (Java development kit). The kit comes with a JRE (Java Runtime environment). Thus the user isn't required to download and install the JRE separately.

Eclipse IDE Installation

Step 1: Go to Eclipse official website and navigate to its download page – Eclipse download. Download Eclipse IDE for Java EE developers. All the recent releases are available on the page.

Make sure you opt and download the appropriate eclipse IDE as per your system configuration. There are two download links available for 64 bit windows operating system and 32-bit windows operating system.

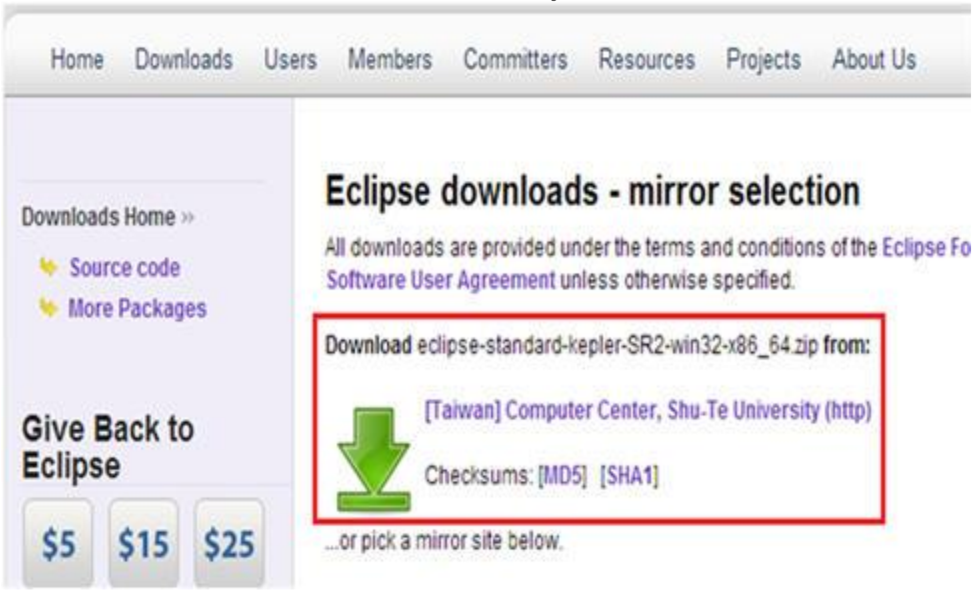

Home Downloads Users Members Committers Resources Projects About Us
Eclipse Downloads
Choose the correct version from here
Packages Java™ 8 Support Developer Builds
Eclipse Kepler (4.3.2) SR2 Packages for Windows
Eclipse Standard 4.3.2, 200 MB
Downloaded 2,669,740 Times Other Downloads
Windows 32 Bit
Windows 64 Bit
Package Solutions
Filter Packages
Eclipse IDE for Java EE Developers, 250 MB
Downloaded 1,440,082 Times
Eclipse IDE for Java Developers, 153 MB
Downloaded 626,241 Times

The Eclipse Platform, and all the tools needed to develop and debug it: Java and Plug-in Development Tooling, Git and CVS...

Tools for Java developers creating Java EE and Web applications, including a Java IDE, tools for Java EE, JPA, JSF, Mylyn...

The essential tools for any Java developer, including a Java IDE, a CVS client, Git client, XML Editor, Mylyn, Maven integration...

Step 2: As soon as we click on the download link, the user is re-directed to the fresh page securing information about the current download. Click on the download icon and you are done.



Eclipse downloads - mirror selection

All downloads are provided under the terms and conditions of the Eclipse Fo Software User Agreement unless otherwise specified.

Download eclipse-standard-kepler-SR2-win32-x86_64.zip from:

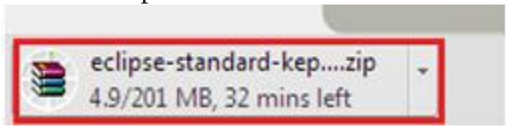
[Taiwan] Computer Center, Shu-Te University (http)

Checksums: [MD5] [SHA1]

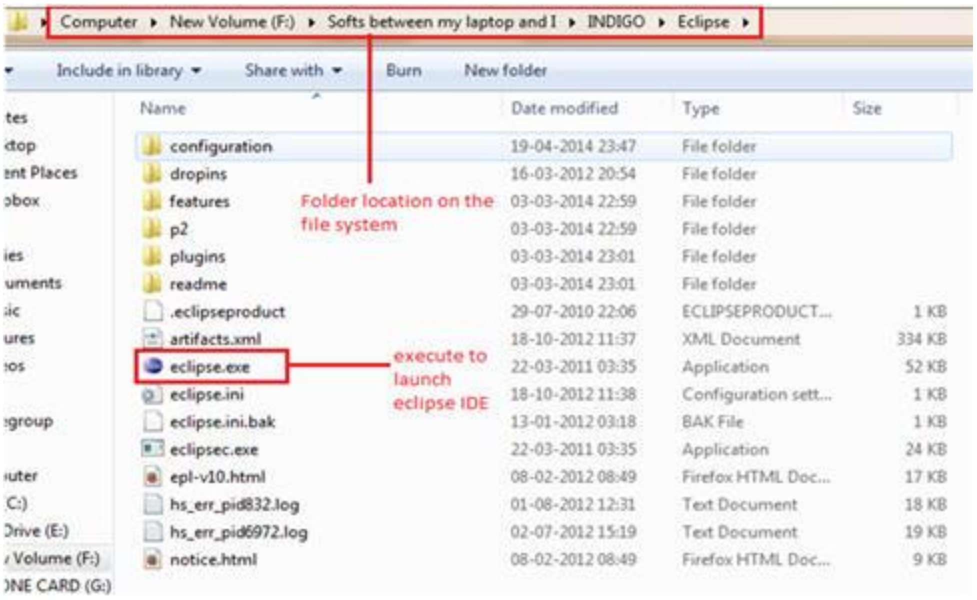
...or pick a mirror site below.

It may take a few minutes before you can download the complete zip folder.

Step 3: Once downloaded, copy the folder and place it in the desired location on your file system.



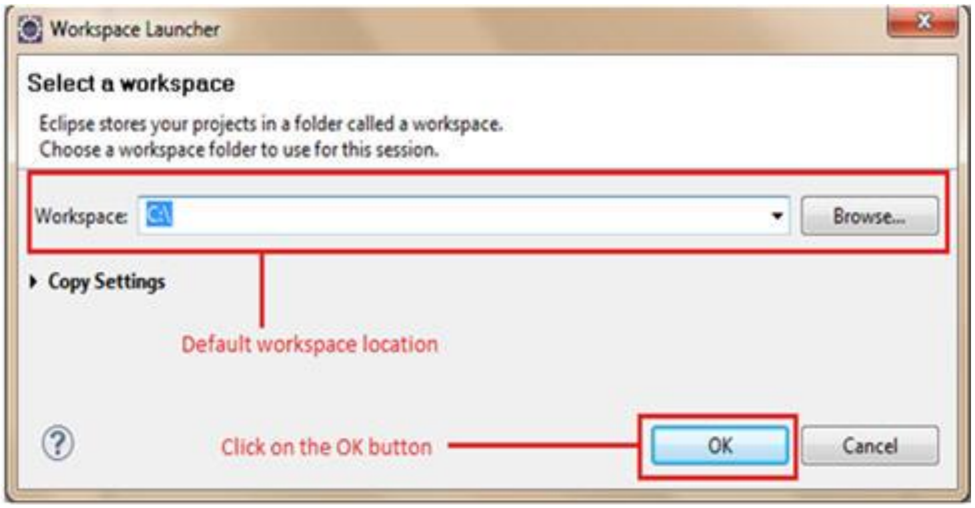
Step 4: Extract the zipped folder, a folder named as eclipse can be seen. The folder embodies all the required application and source files.


Computer ▸ New Volume (F:) ▸ Softs between my laptop and I ▸ INDIGO ▸ Eclipse ▸
Include in library
Share with
Burn
New folder
Folder location on the file system
execute to launch eclipse IDE

Name	Date modified	Type	Size
configuration	19-04-2014 23:47	File folder	
dropins	16-03-2012 20:54	File folder	
features	03-03-2014 22:59	File folder	
p2	03-03-2014 22:59	File folder	
plugins	03-03-2014 23:01	File folder	
readme	03-03-2014 23:01	File folder	
.eclipseproduct	29-07-2010 22:06	ECLIPSEPRODUCT...	1 KB
artifacts.xml	18-10-2012 11:37	XML Document	334 KB
eclipse.exe	22-03-2011 03:35	Application	52 KB
eclipse.ini	18-10-2012 11:38	Configuration sett...	1 KB
eclipse.ini.bak	13-01-2012 03:18	BAK File	1 KB
eclipsec.exe	22-03-2011 03:35	Application	24 KB
epl-v10.html	08-02-2012 08:49	Firefox HTML Doc...	17 KB
hs_err_pid832.log	01-08-2012 12:31	Text Document	18 KB
hs_err_pid6972.log	02-07-2012 15:19	Text Document	19 KB
notice.html	08-02-2012 08:49	Firefox HTML Doc...	9 KB

Step 5: Launch the eclipse IDE using "eclipse.exe" residing inside the eclipse folder. Refer the above illustration for the same.

Step 6: The application will prompt you to specify the workspace location. Workspace is that location where all your eclipse projects will be residing. Enter/Browse the desired location or the user can simply opt for the default location and click on the OK button.



Select a workspace

Eclipse stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

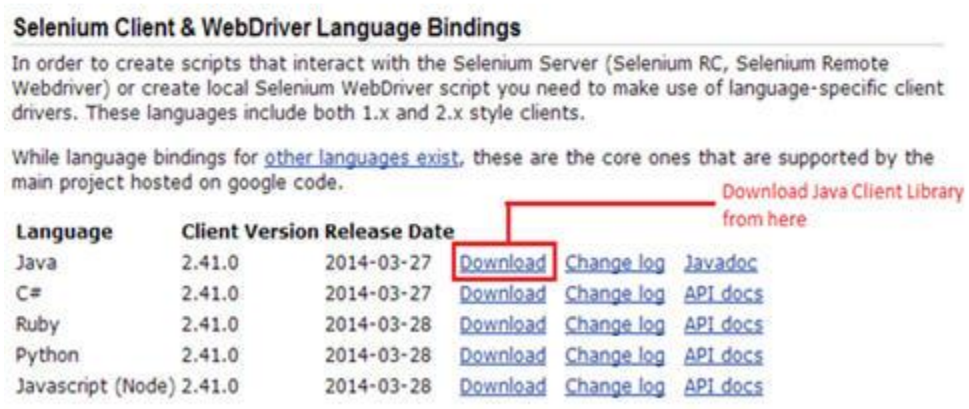
Workspace: C:\

Configuring WebDriver

As we would be using Java as the programming language for this series and in order to create test scripts in java, we would have to introduce language- specific client drivers. Thus, let us begin with the downloading of Selenium Java Client Libraries.

Download the Selenium Java Client Libraries

Step 1: Go to Selenium's official website and navigate to its download page – "http://docs.seleniumhq.org/download/". Refer the section in the below illustration where you can find Client Libraries listed for distinct programming languages. Click on the download link for Java Client Library.


Selenium Client & WebDriver Language Bindings

In order to create scripts that interact with the Selenium Server (Selenium RC, Selenium Remote Webdriver) or create local Selenium WebDriver script you need to make use of language-specific client drivers. These languages include both 1.x and 2.x style clients.

While language bindings for other languages exist, these are the core ones that are supported by the main project hosted on google code.

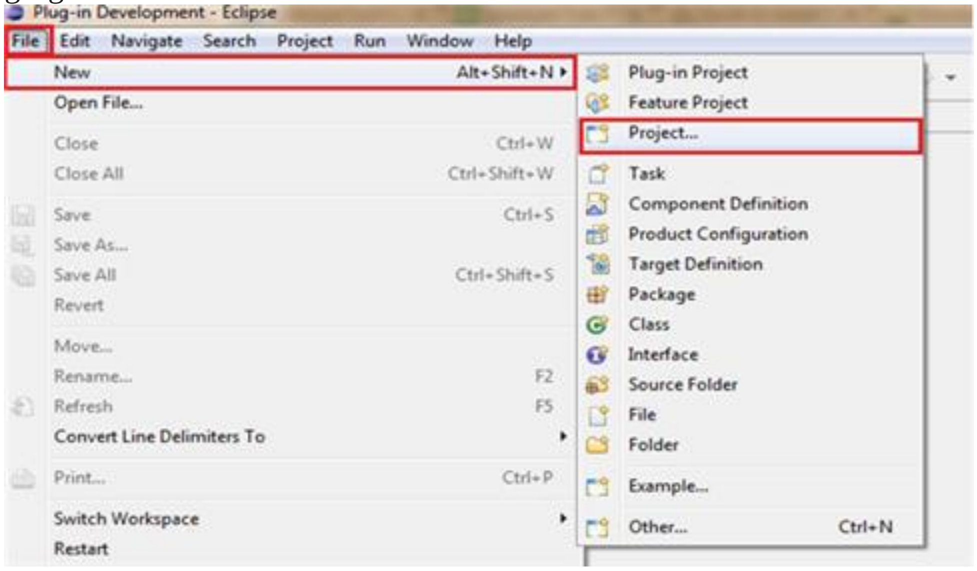
Language	Client Version	Release Date			
Java	2.41.0	2014-03-27	Download	Change log	Javadoc
C#	2.41.0	2014-03-27	Download	Change log	API docs
Ruby	2.41.0	2014-03-28	Download	Change log	API docs
Python	2.41.0	2014-03-28	Download	Change log	API docs
Javascript (Node)	2.41.0	2014-03-28	Download	Change log	API docs

It may take a few minutes before you can download the complete zipped folder.

Step 2: Once downloaded, copy the folder and place it in the desired location on your file system.
Step 3: Extract the zipped folder, a folder named as “Selenium-2.41.0.zip“can be seen. The folder embodies all the required jar files which enable users to create test scripts in Java.
Thus these libraries can be configured in Eclipse IDE.

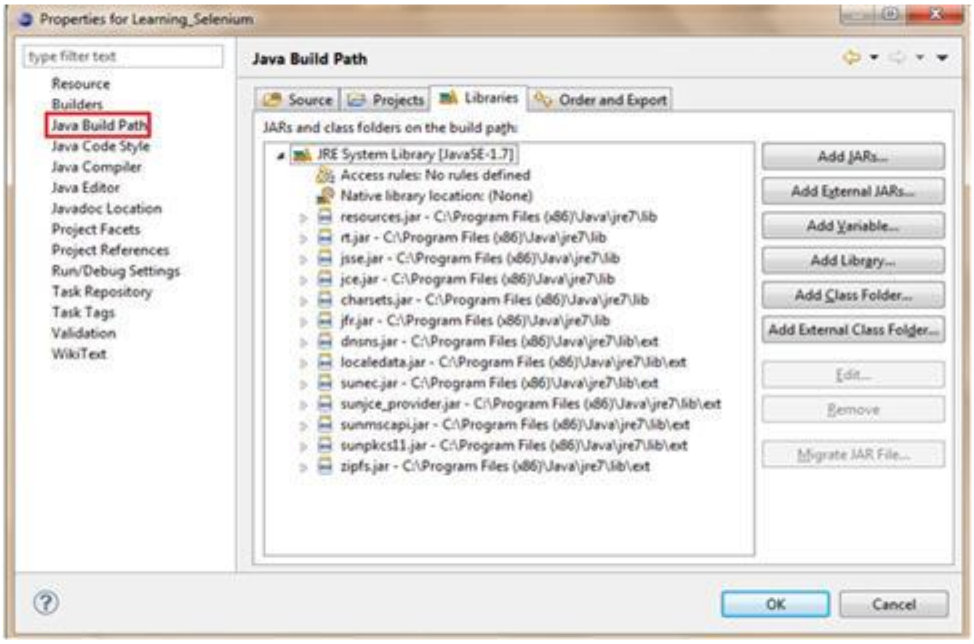
Configuring Libraries with Eclipse IDE

Step 1: Navigate towards Eclipse IDE. Create a new java based project following File -> New -> Java Project. Refer the following figure for the same.



Step 2: Provide a user defined name for your Java Project. Let us provide the name as Learning_Selenium and Click on the Finish Button. The newly created project can be viewed at the left side of the screen in the package explorer panel.
Step 3: Create a new Java class named as “First_WebdriverClass” under the source folder by right clicking on it and navigating to New -> class.

Step 4: Now let us configure the libraries into our Java project. For this, select the project and Right click on it. Select "Properties" within the listed options. The following screen appears, Select "Java Build Path" from the options.

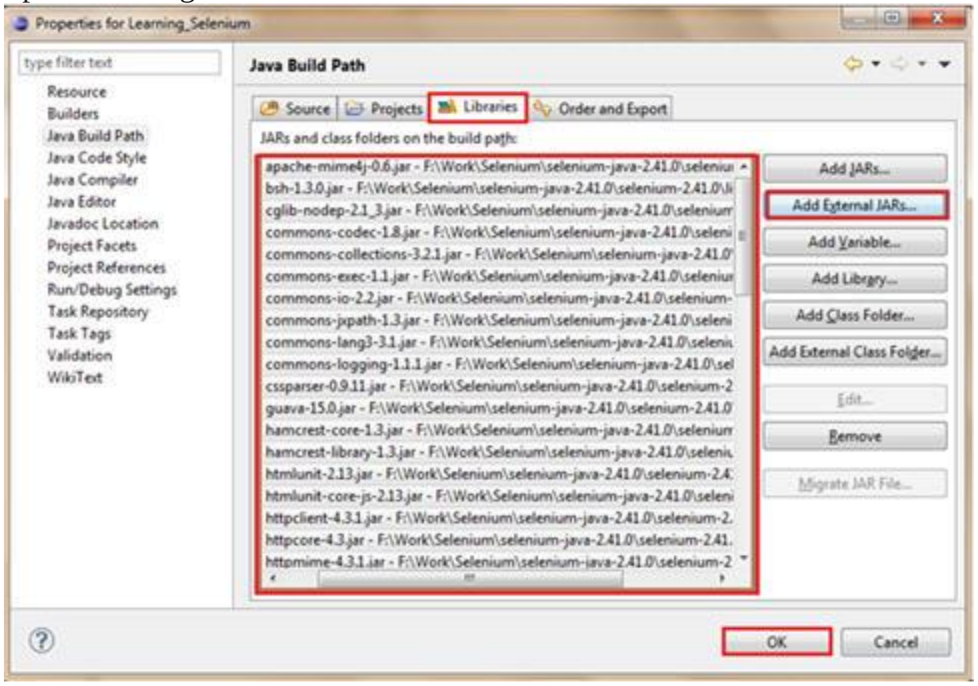


JARs and class folders on the build path:

```
JRE System Library [JavaSE-1.7]
    Access rules: No rules defined
    Native library location: (None)
    resources.jar - C:\Program Files (x86)\Java\jre7\lib
    rt.jar - C:\Program Files (x86)\Java\jre7\lib
    jsse.jar - C:\Program Files (x86)\Java\jre7\lib
    jce.jar - C:\Program Files (x86)\Java\jre7\lib
    charsets.jar - C:\Program Files (x86)\Java\jre7\lib
    jfr.jar - C:\Program Files (x86)\Java\jre7\lib
    dnsns.jar - C:\Program Files (x86)\Java\jre7\lib\ext
    localedata.jar - C:\Program Files (x86)\Java\jre7\lib\ext
    sunec.jar - C:\Program Files (x86)\Java\jre7\lib\ext
    sunjce_provider.jar - C:\Program Files (x86)\Java\jre7\lib\ext
    sunmscapi.jar - C:\Program Files (x86)\Java\jre7\lib\ext
    sunpkcs11.jar - C:\Program Files (x86)\Java\jre7\lib\ext
    zipfs.jar - C:\Program Files (x86)\Java\jre7\lib\ext
```

Step 5: By default, "Libraries" tab is opened. If not, click on the "Libraries" tab. Then, click on the "Add External Jars…" button. Browse to the location where we have saved the extracted folder for Java Client Libraries.

Step 6: Select all the JAR files present in the "selenium-java-2.41.0" folder and click on open button within the dialog box. The properties dialog box should look like the below illustration.

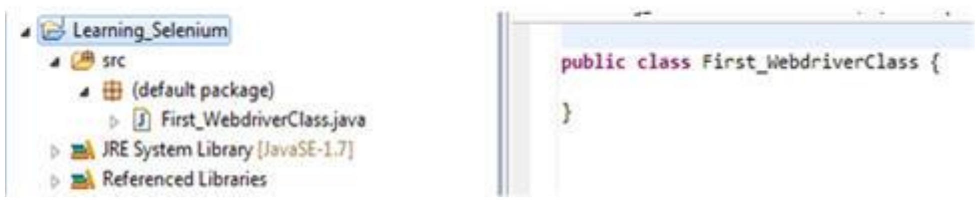


JARs and class folders on the build path:

```
apache-mime4j-0.6.jar - F:\Work\Selenium\selenium-java-2.41.0\seleniu
bsh-1.3.0.jar - F:\Work\Selenium\selenium-java-2.41.0\selenium-2.41.0\li
cglib-nodep-2.1_3.jar - F:\Work\Selenium\selenium-java-2.41.0\selenium
commons-codec-1.8.jar - F:\Work\Selenium\selenium-java-2.41.0\seleni
commons-collections-3.2.1.jar - F:\Work\Selenium\selenium-java-2.41.0
commons-exec-1.1.jar - F:\Work\Selenium\selenium-java-2.41.0\seleniu
commons-io-2.2.jar - F:\Work\Selenium\selenium-java-2.41.0\selenium-
commons-jxpath-1.3.jar - F:\Work\Selenium\selenium-java-2.41.0\seleni
commons-lang3-3.1.jar - F:\Work\Selenium\selenium-java-2.41.0\seleni
commons-logging-1.1.1.jar - F:\Work\Selenium\selenium-java-2.41.0\sel
cssparser-0.9.11.jar - F:\Work\Selenium\selenium-java-2.41.0\selenium-2
guava-15.0.jar - F:\Work\Selenium\selenium-java-2.41.0\selenium-2.41.0
hamcrest-core-1.3.jar - F:\Work\Selenium\selenium-java-2.41.0\selenium
hamcrest-library-1.3.jar - F:\Work\Selenium\selenium-java-2.41.0\seleni
htmlunit-2.13.jar - F:\Work\Selenium\selenium-java-2.41.0\selenium-2.4
htmlunit-core-js-2.13.jar - F:\Work\Selenium\selenium-java-2.41.0\seleni
httpclient-4.3.1.jar - F:\Work\Selenium\selenium-java-2.41.0\selenium-2.
httpcore-4.3.jar - F:\Work\Selenium\selenium-java-2.41.0\selenium-2.41.
httpmime-4.3.1.jar - F:\Work\Selenium\selenium-java-2.41.0\selenium-2
```

Step 7: Click on the "OK" button within the dialog box so as to complete the configuration part of Selenium Libraries in our java project.

The project will look like the following:



Learning_Selenium
src
(default package)
First_WebdriverClass.java
JRE System Library [JavaSE-1.7]
Referenced Libraries

```
public class First_WebdriverClass {

}
```

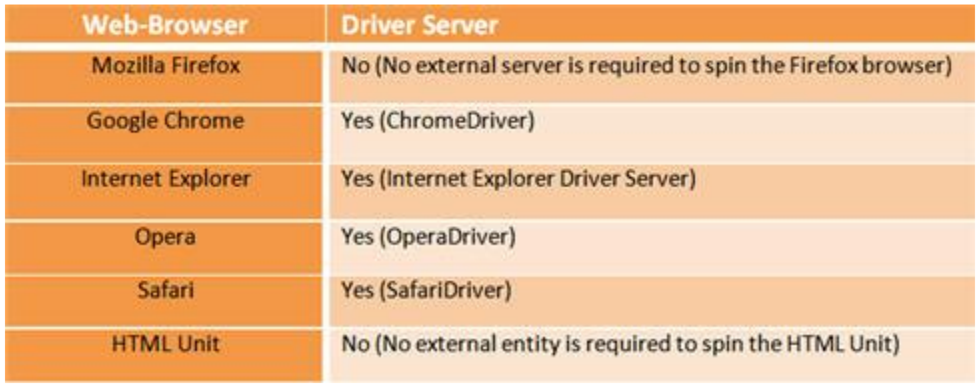
Available Drivers

There are a number of driver classes available in WebDriver, each catering a specific web browser. Each browser has a different driver implementation in WebDriver.

In WebDriver, a few of the browsers can be automated directly where as some of the web browsers require an external entity to be able to automate and execute the test script. This external entity is known as Driver Server. Thus, user is required to download the Driver Server for different web browsers.

Notice that there is a separate Driver Server for each of the web browser and user cannot use one Driver Server for web browsers other than the one it is designated for.

Below is the list of available web browsers and their corresponding Server Drivers.



Web-Browser	Driver Server
Mozilla Firefox	No (No external server is required to spin the Firefox browser)
Google Chrome	Yes (ChromeDriver)
Internet Explorer	Yes (Internet Explorer Driver Server)
Opera	Yes (OperaDriver)
Safari	Yes (SafariDriver)
HTML Unit	No (No external entity is required to spin the HTML Unit)

Conclusion

In this tutorial, we accustomed you with all the environment setup and installation to be done prior to creation of WebDriver test scripts.

Here are the cruxes of this article.

- Prior to the creation of WebDriver based test scripts, few utilities and packages are required to be installed.
- Install JDK (Java Development Kit). Remember, the user is not supposed to install JRE separately because it is distributed bundled with the kit.

- Download Eclipse IDE. User is only required to download the package and he/she is good to go. No other installation is required with Eclipse.
- Download Java Client Libraries to be able to create test script in java programming language.
- Launch eclipse using eclipse.exe. Select the workspace where you would want to save the projects.
- Create a new java project in the eclipse. Create a new java class within the project.
- Configure the eclipse by importing jars files for Java Client Drivers.
- In WebDriver, a few of the browsers can be automated directly where as some of the web browsers require an external Driver Server.
- Firefox and HTML Unit are the only browsers that cannot be automated directly. Thus they do not require any separate Driver Server. All other commonly known web browsers like Chrome, Safari, Internet Explorer etc. requires Driver Servers.

Next Tutorial #10 => Now that we are done with the entire setup and installation, in the next tutorial **we would create our own WebDriver test script using Java.**

A remark for the readers: While our next tutorial of the ***Selenium tutorials series*** is in the processing mode, install the packages mentioned in this tutorial and the required utilities to get started. Most of the WebDriver related packages can be found at the Selenium's official website.

Let us know if you face any issues in installation process.

Implementation of Our First WebDriver Script – Selenium WebDriver Tutorial #10

In the previous two tutorials, we made you acquainted with the basic architecture and features of WebDriver and the infrastructure required to get started with Selenium WebDriver. Assuming that you all might have set up the system with all the necessary utilities and packages, ***we will move further with the implementation of our first WebDriver test script.***
Therefore, motioning ahead with the consequent ***Selenium WebDriver tutorial***, we would be creating WebDriver script. We would also scatter the light on the basic and commonly used **WebDriver commands**. We would also learn about the **locating strategies of UI elements** and their inclusion in the test scripts. We would also study Get Commands in the detail.

Script Creation

For script creation, we would be using "Learning_Selenium" project created in the previous tutorial and "gmail.com" as the application under test (AUT).

Scenario:

- Launch the browser and open "Gmail.com".
- Verify the title of the page and print the verification result.
- Enter the username and Password.
- Click on the Sign in button.
- Close the web browser.

Step 1: Create a new java class named as "Gmail_Login" under the "Learning_Selenium" project.
Step 2: Copy and paste the below code in the "Gmail_Login.java" class.

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;

public class Gmail_Login {
/**
* @param args
*/
       public static void main(String[] args) {

// objects and variables instantiation
           WebDriver driver = new FirefoxDriver();
           String appUrl = "https://accounts.google.com";

// launch the firefox browser and open the application url
        driver.get(appUrl);

// maximize the browser window
        driver.manage().window().maximize();

// declare and initialize the variable to store the expected title of the webpage.
        String expectedTitle = " Sign in - Google Accounts ";

// fetch the title of the web page and save it into a string variable
        String actualTitle = driver.getTitle();

// compare the expected title of the page with the actual title of the page and print the result
        if (expectedTitle.equals(actualTitle))
        {
            System.out.println("Verification Successful - The correct title is displayed on the web page.");
        }
        else
        {
            System.out.println("Verification Failed - An incorrect title is displayed on the web page.");
        }

// enter a valid username in the email textbox
        WebElement username = driver.findElement(By.id("Email"));
        username.clear();
        username.sendKeys("TestSelenium");

// enter a valid password in the password textbox
        WebElement password = driver.findElement(By.id("Passwd"));
        password.clear();
        password.sendKeys("password123");

// click on the Sign in button
        WebElement SignInButton = driver.findElement(By.id("signIn"));
        SignInButton.click();

// close the web browser
        driver.close();
        System.out.println("Test script executed successfully.");

// terminate the program
        System.exit(0);
    }
}
```

The above code is equivalent to the textual scenario presented earlier.

Code Walkthrough

Import Statements:

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.By;
```

Prior to the actual scripting, we need to import the above packages:

import *org.openqa.selenium.WebDriver* – References the WebDriver interface which is required to instantiate a new web browser.

import *org.openqa.selenium.firefox.FirefoxDriver* – References the FirefoxDriver class that is required instantiate a Firefox specific driver on the browser instance instantiated using WebDriver interface.

import *org.openqa.selenium.WebElement* – References to the WebElement class which is required to instantiate a new web element.

import *org.openqa.selenium.By* – References to the By class on which a locator type is called.

As and when our project would grow, it is evident and logical that we might have to introduce several other packages for more complex and distinct functionalities like excel manipulations, database connectivity, logging, assertions etc.

Object Instantiation

WebDriver driver = new FirefoxDriver();

We create a reference variable for WebDriver interface and instantiate it using FirefoxDriver class. A default Firefox profile will be launched which means that no extensions and plugins would be loaded with the Firefox instance and that it runs in the safe mode.

Launching the Web browser

driver.get(appUrl);

A *get()* method is called on the WebDriver instance to launch a fresh web browser instance. The string character sequence passed as a parameter into the *get()* method redirects the launched web browser instance to the application URL.

Maximize Browser Window

driver.manage().window().maximize();

The *maximize()* method is used to maximize the browser window soon after it is re-directed to the application URL.

Fetch the page Title

driver.getTitle();

The *getTitle()* method is used to fetch the title of the current web page. Thus, the fetched title can be loaded to a string variable.

Comparison between Expected and Actual Values:

```
if (expectedTitle.equals(actualTitle))
        {
            System.out.println("Verification Successful - The correct title is displayed on the web page.");
        }
        else
        {
            System.out.println("Verification Failed - An incorrect title is displayed on the web page.");
        }
```

The above code uses the conditional statement java constructs to compare the actual value and the expected value. Based on the result obtained, the print statement would be executed.

WebElement Instantiation

WebElement username = driver.findElement(By.id("Email"));

In the above statement, we instantiate the WebElement reference with the help of *"driver.findElement(By.id("Email"))"*. Thus, username can be used to reference the Email textbox on the user interface every time we want to perform some action on it.

Clear Command

username.clear();

The clear() method/command is used to clear the value present in the textbox if any. It also clears the default placeholder value.

sendKeys Command

username.sendKeys("TestSelenium ");

The *sendKeys()* method/command is used to enter/type the specified value (within the parentheses) in the textbox. Notice that the *sendKeys()* method is called on the WebElement object which was instantiated with the help of element property corresponding to the UI element.

The above block of code enters the string "TestSelenium" inside the Email textbox on the Gmail application.

sendKeys is one of the most popularly used commands across the WebDriver scripts.

Click Command

SignInButton.click();

Like *sendKeys(), click()* is another excessively used command to interact with the web elements. *Click()* command/method is used to click on the web element present on the web page.

The above block of code clicks on the "Sign in" button present on the Gmail application.

Notes:

- Unlike sendKeys() method, click() methods can never be parameterized.
- At times, clicking on a web element may load a new page altogether. Thus to sustain such cases, click() method is coded in a way to wait until the page is loaded.

<u>Close the Web Browser</u>

driver.close();

The close() is used to close the current browser window.

<u>Terminate the Java Program</u>

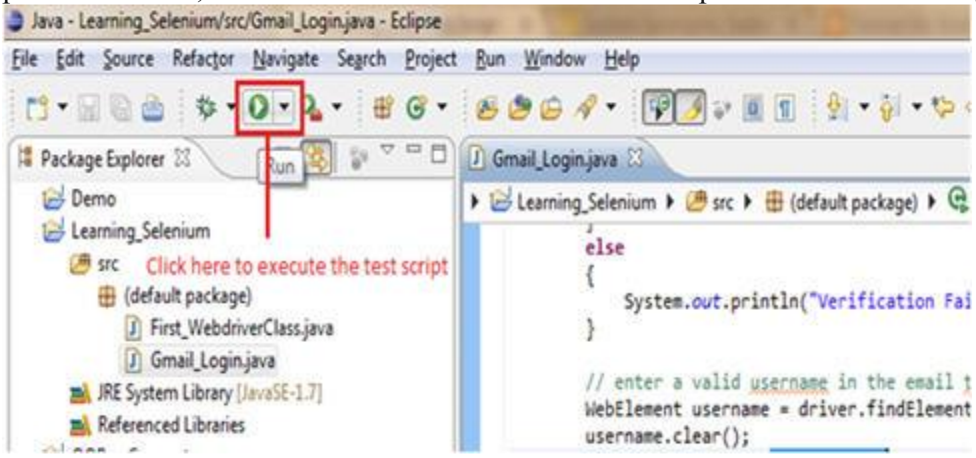
System.exit(0);

The Exit() method terminates the Java program forcefully. Thus, remember to close all the browser instances prior terminating the Java Program.

Test Execution

The test script or simply the java program can be executed in the following ways:

#1. Under the Eclipse's menu bar, there is an icon to execute the test script. Refer the following figure.


Java - Learning_Selenium/src/Gmail_Login.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Run
Demo
Learning_Selenium
src
Click here to execute the test script
(default package)
First_WebdriverClass.java
Gmail_Login.java
JRE System Library [JavaSE-1.7]
Referenced Libraries
Gmail_Login.java
Learning_Selenium ▸ src ▸ (default package) ▸

```
}
else
{
    System.out.println("Verification Fai
}

// enter a valid username in the email t
WebElement username = driver.findElement
username.clear();
```

Make a note that only the class which is selected would be executed.

#2. Right click anywhere inside the class within the editor, select "Run As" option and click on the "Java Application".

#3. Another shortcut to execute the test script is – Press ctrl + F11.

At the end of the execution cycle, the print statement "Test script executed successfully." can be found in the console.

Locating Web Elements

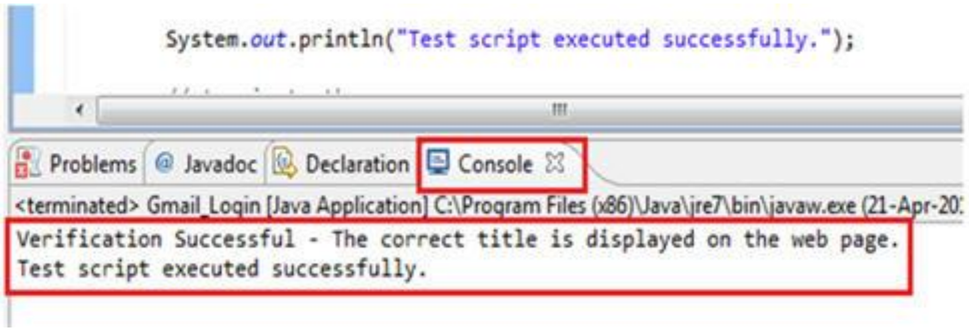
Web elements in WebDriver can be located and inspected in the same way as we did in the previous tutorials of Selenium IDE. Selenium IDE and Firebug can be used to inspect the web element on the GUI. It is highly suggested to use Selenium IDE to find the web elements. Once the web element is successfully found, copy and paste the target value within the WebDriver code. The types of locators and the locating strategies are pretty much the same except for the syntax and their application.

In WebDriver, web elements are located with the help of the dynamic finders (findElement(By.locatorType(“locator value”))).

Locator Type	Syntax	Description
id	driver.findElement (By.id(“ID_of_Element”))	Locate by value of the “id” attribute
className	driver.findElement (By.className (“Class_of_Element”))	Locate by value of the “class” attribute
linkText	driver.findElement (By.linkText(“Text”))	Locate by value of the text of the hyperlink
partialLinkText	driver.findElement (By.partialLinkText (“PartialText”))	Locate by value of the sub-text of the hyperlink
name	driver.findElement (By.name (“Name_of_Element”))	Locate by value of the “name” attribute
xpath	driver.findElement (By.xpath(“Xpath”))	Locate by value of the xpath
cssSelector	driver.findElement (By.cssSelector (“CSS Selector”))	Locate by value of the CSS selector
tagName	driver.findElement (By.tagName(“input”))	Locate by value of its tag name

Sample Code:

driver.findElement(By.id(“Email”));


```
System.out.println("Test script executed successfully.");
```

```
<terminated> Gmail_Login [Java Application] C:\Program Files (x86)\Java\jre7\bin\javaw.exe (21-Apr-20
Verification Successful - The correct title is displayed on the web page.
Test script executed successfully.
```

Locator Types and their Syntax
Conclusion

In this tutorial, we developed an automation script using WebDriver and Java. We also discussed the various components that constitute a WebDriver script.

<u>Here are the cruxes of this Selenium WebDriver Tutorial:</u>

- Prior to the actual scripting, we need to import a few packages to be able to create a WebDriver script.
  - ***import****openqa.selenium.By;*
  - ***import****openqa.selenium.WebDriver;*
  - ***import****openqa.selenium.WebElement;*
  - ***import****openqa.selenium.firefox.FirefoxDriver;*
- A *get()* method used to launch a fresh web browser instance. The character sequence passed as a parameter into the get() method redirects the launched web browser instance to the application URL.
- The *maximize()* method is used to maximize the browser window.
- The *clear()* method is used to clear the value present in the textbox if any.
- The *sendKeys()* method is used to enter the specified value in the textbox.
- *Click()* method is used to click on the web element present on the web page.
- In WebDriver, web elements can be located using Dynamic finders.
- The following are the available locator types:
  - id
  - className
  - name
  - xpath
  - cssSelector
  - linkText
  - partialLinkText
  - tagName

Moving ahead, in the next tutorial, we would shift our focus towards a framework that aids to Automation testing known as TestNG. We would have a detailed study on the various kinds of the annotations provided by the framework.

Next tutorial #11: Before diving deep into Frameworks we will see details about JUnit – an open source unit testing tool. Most of the programmers use JUnit as it is easy and does not take much effort to test. This tutorial will give an insight about JUnit and its usage in selenium script.

A remark for the readers: While our next tutorial of the Selenium series is in the processing mode, readers can start creating their own basic WebDriver scripts. For more advance scripts and concepts, we will have various other Selenium WebDriver tutorials coming up in this series.

Let us know in comments if you have any problem creating or executing the WebDriver scripts.

Introduction to JUnit Framework and Its Usage in Selenium Script – Selenium Tutorial #11

This tutorial will give an insight about JUnit and its usage in selenium script. This is tutorial #11 in our comprehensive Selenium tutorials series.

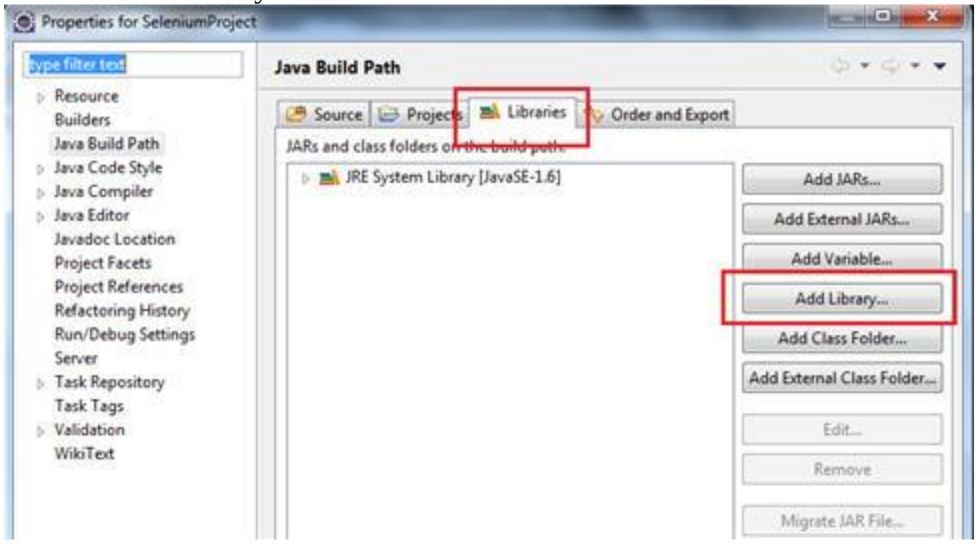
Basically JUnit is an open source unit testing tool and used to test small/large units of code. To run the JUnit test you don't have to create class object or define main method. JUnit provide assertion library which is used to evaluate the test result. Annotations of JUnit are used to run the test method. JUnit is also used to run the Automation suite having multiple test cases.

Adding JUnit library in Java project

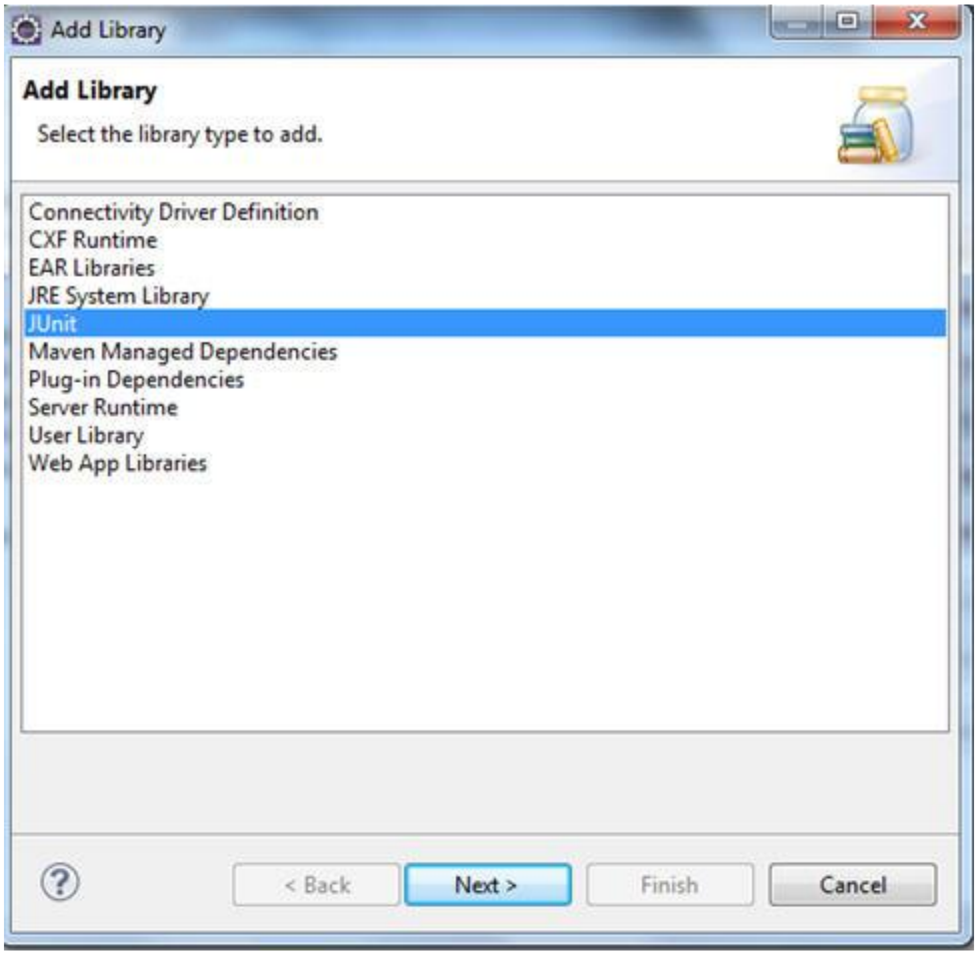
First we will learn how to add JUnit library in your Java project:

Step #1: Right click on Java project->Build Path->Configure Build path

Step #2: Click Libraries->Add Library



Step #3: Click on Junit.

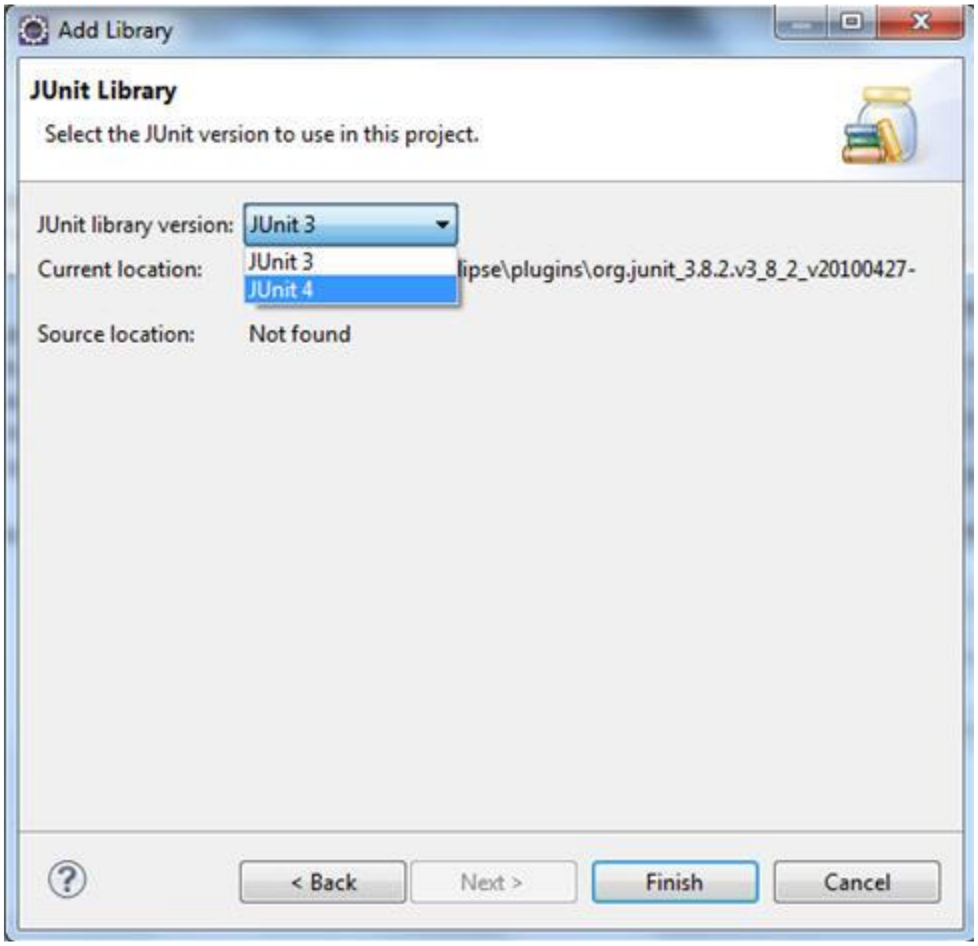

Add Library

Add Library

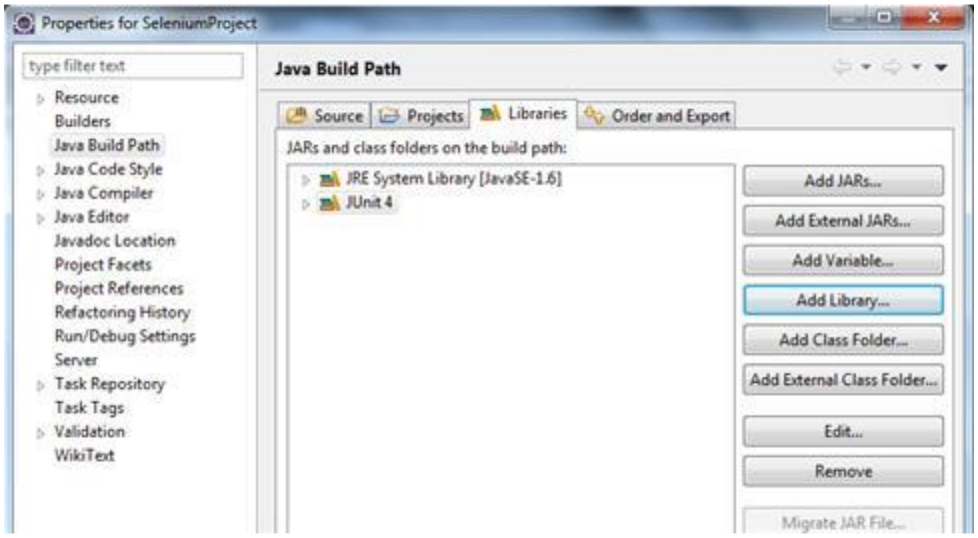
Select the library type to add.

Connectivity Driver Definition
CXF Runtime
EAR Libraries
JRE System Library
JUnit
Maven Managed Dependencies
Plug-in Dependencies
Server Runtime
User Library
Web App Libraries

Step #4: Select Junit4->Finish


Add Library
JUnit Library
Select the JUnit version to use in this project.
JUnit library version: JUnit 3
JUnit 3
JUnit 4
Current location: lipse\plugins\org.junit_3.8.2.v3_8_2_v20100427-
Source location: Not found
< Back
Next >
Finish
Cancel

Step #5: Click OK.



There are many frameworks like Data Driven Framework, Keyword Driven Framework, and Hybrid Framework which use Junit tool as test runner and which will help to start the batch execution and reporting.

JUnit Annotations Used in Selenium scripts

There are many annotations available in Junit. Here we have described few annotations which are used very frequently in Selenium scripts and framework.

#1. @Test

@Test annotation is used to run a Junit test.

Example:

```
@Test
public void junitTest()
{
System.out.println("Running Junit test");
Assert.assertEquals(1,1);
}
```

How to Run a Junit test:

Navigate to run ->Run as Junit test

#2. @Before:

@Before annotation is used to run any specific test before each test.

```
public class Junttest {
@Before
public void beforeTest(){
System.out.println("Running before test");
}

@Test
public void junitTest(){
 System.out.println("Running Junit test");
}
}
```

Output:

Running before test

Running Junit test

Example of before annotation using two junit test method.

```
public class Junttest {
@Before
public void beforeTest(){
System.out.println("Running before test");
}

@Test
public void junitTest(){
 System.out.println("Running Junit test");
}

@Test
public void secondJunitTest(){
System.out.println("Running second Junit test");
}
}
```

Output:

```
Running before test
Running Junit test
Running before test
Running second Junit test
```

Before running junitTest method beforeTest method will run. Similarly before running secondJuntiTest again beforeTest method will run and produces output like above.

#3. @BeforeClass

This method executes once before running all test. The method has to be a static method. Initialization of properties files, databases etc are done in beforeClass method.

```
public class Junttest {
@BeforeClass
public static void beforeClassTest(){
System.out.println("Executed before class method");
}

@Test
public void junitTest(){
 System.out.println("Running Junit test");
}

@Test
public void secondJunitTest(){
System.out.println("Running second Junit test");
}
}
```

Output:

```
Executed before class method
Running Junit test
Running second Junit test
```

#4. @After

This method executes after each test.

```
public class Junttest {
@Test
public void junitTest(){
System.out.println("Running Junit test");
}

@After
public void afterTest(){
 System.out.println("Running after method");
}
}
```

Output:

```
Running Junit test
Running after method
```

#5. @AfterClass

Like @BeforeClass, @AfterClass executes once after executing all test methods. Like @BeforeClass method, @AfterClass method has to be a static method.

```
public class Junttest {

@Test
public void junitTest(){
System.out.println("Running Junit test");
}

@Test
public void secondJunitTest(){
                        System.out.println("Running second Junit test");
}

@AfterClass
Public static void afterClassTest(){
System.out.println("Running afterclass method");
}
}
```

Output:

```
Running Junit test
Running second Junit test
Running afterclass method
```

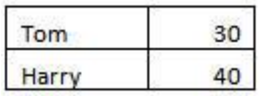
Junit assertions are used to validate certain condition and stops execution of program if the conditions are not satisfied.

#6. Parameterized Junit class:

Parameterized class is used to run same scenario with multiple dataset.

Below is the example to pass multiple parameters in a Junit test.

@Parameters annotation tag is used to pass multiple data. Here, we have taken 2*2 dimensional array and the data can be visualized like below:



Tom	30
Harry	40

```
@RunWith(Parameterized.class)
public class Junttest {
public String name;
public int age;
public Junttest(String name,int age){
this.name=name;
this.age=age;
}

@Test
public void testMethod(){
System.out.println("Name is: "+name +" and age is: "+age);
}

@Parameters
public static Collection<Object[]> parameter(){
Object[][] pData=new Object[2][2];
pData[0][0]="Tom";
pData[0][1]=30;
pData[1][0]="Harry";
pData[1][1]=40;
return Arrays.asList(pData);
}
}
```

JUnit Assertions

JUnit assertEquals: This checks if the two values are equal and assertion fails if both values are not equal. This compares Boolean, int, String, float, long, char etc.

Syntax:

Assert.assertEqual("excepted value", "actual value");

Example:

Assert.assertEqual("ABC","ABC"); //Both the strings are equal and assertion will pass.

Assert.assertEqual("ABC","DEF"); //Assertion will fail as both the strings are not equal.

Assert.assertEqual("Strings are not equal", "ABC","DEF"); //message will be thrown if equal condition is not satisfied.

Below is the example of use of JUnit assertion in selenium:

```
1 String username=driver.findElement(By.id("username")).getText();
2 String password=driver.findElement(By.id("password")).getText();
3 Assert.assertEqual("Mismatch in both the string", username, password);
```

In above example assertion will fail as both the strings are not equal. One is text of username field and other is the text of password field.

JUnit assertTrue: Returns true if the condition is true and assertion fails if the condition is false.

Assert.assertTrue("message", condition);

Assert.assertTrue("Both the strings are not equal", ("HelloWorld").equals("HelloWorld"));

Here assertion will pass as both the strings match. It will print message if the assertion fails.

JUnit assertFalse: Returns true if the condition is false and assertion fails if the condition is true.

Assert.assertFalse("message", condition);

Assert.assertFalse("Both the strings are equal", ("Hello").equals("HelloWorld"));

There will not be any assertion error as the condition is false.

Conclusion:

Most of the programmers use Junit as it is easy and does not take much effort to test. A simple green or red bar will show the actual result of the test. Junit makes life easy as it has its own set of libraries and annotations. Here we have also described commonly used annotations used with selenium scripts and framework.

More detail about framework and use of Junit annotations will be discussed in upcoming tutorial which is dedicated exclusively for framework design using Junit. This tutorial will help us in designing the framework using Junit.

Next Tutorial #12: In next tutorial we would discuss all about TestNG, its features and its applications. TestNG is an advance framework designed in a way to leverage the benefits by both the developers and testers.

How to Use TestNG Framework for Creating Selenium Scripts – TestNG Selenium Tutorial #12

In the last few tutorials, we shed light on the basic and commonly used WebDriver commands. We also learned about the locating strategies of UI elements and their inclusion in the test scripts. And therefore, we developed our very first WebDriver Automation Test Script.

Moving ahead with this tutorial, we would discuss all about TestNG, its features and its applications.

TestNG is an advance framework designed in a way to leverage the benefits by both the developers and testers. For people already using JUnit, TestNG would seem no different with some advance features. With the commencement of the frameworks, JUnit gained an enormous popularity across the Java applications, Java developers and Java testers, with remarkably increasing the code quality.

See also => **JUnit Tutorial and its usage in Selenium scripts**

Despite being an easy to use and straightforward framework, JUnit has its own limitations which give rise to the need of bringing TestNG into the picture. TestNG was created by an acclaimed programmer named as "Cedric Beust". TestNG is an open source framework which is distributed under the Apache software License and is readily available for download.

Talking about our requirement to introduce TestNG with WebDriver is that it provides an efficient and effective test result format that can in turn be shared with the stake holders to have a glimpse on the product's/application's health thereby eliminating the drawback of WebDriver's incapability to generate test reports. TestNG has an inbuilt exception handling mechanism which lets the program to run without terminating unexpectedly.

Both TestNG and JUnit belong to the same family of Unit Frameworks where TestNG is an extended version to JUnit and is more extensively used in the current testing era.

Features of TestNG

- Support for annotations
- Support for parameterization
- Advance execution methodology that do not require test suites to be created
- Support for Data Driven Testing using Dataproviders
- Enables user to set execution priorities for the test methods
- Supports threat safe environment when executing multiple threads
- Readily supports integration with various tools and plug-ins like build tools (Ant, Maven etc.), Integrated Development Environment (Eclipse).

- Facilitates user with effective means of Report Generation using ReportNG

<u>TestNG versus JUnit</u>

There are various advantages that make TestNG superior to JUnit. Some of them are:

- Advance and easy annotations
- Execution patterns can be set
- Concurrent execution of test scripts
- Test case dependencies can be set

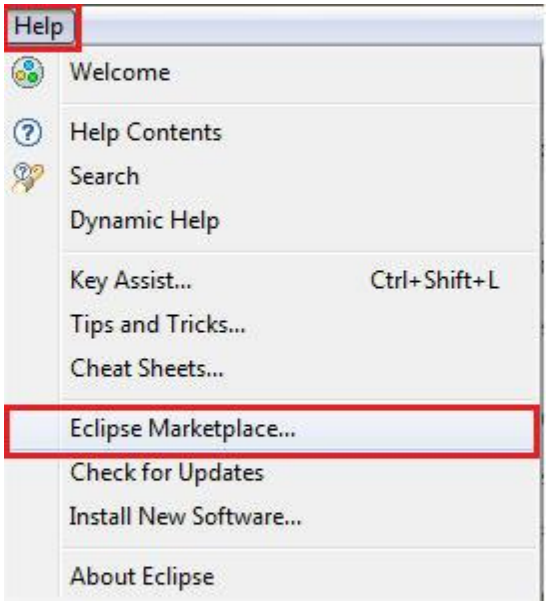
Annotations are preceded by a "@" symbol in both TestNG and JUnit.

So now let us get started with the installation and implementation part.

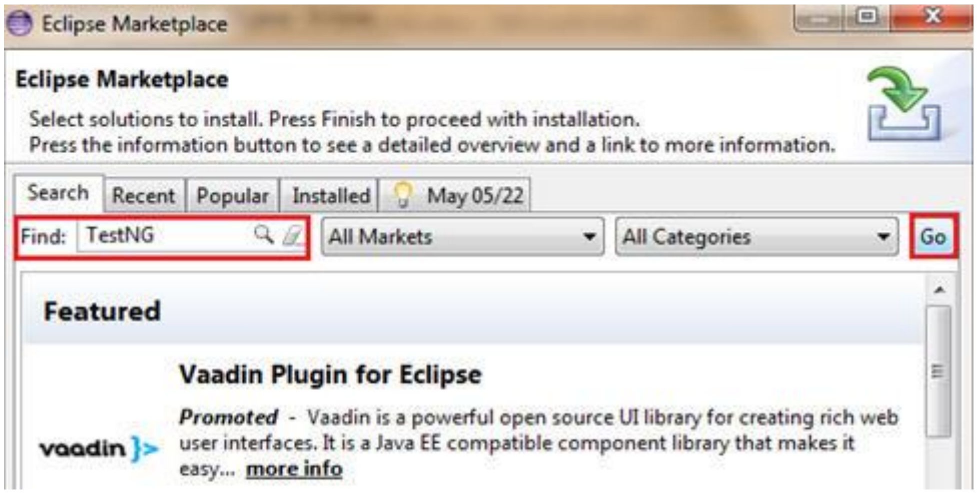
TestNG Installation in Eclipse

Follow the below steps to TestNG Download and installation on eclipse:

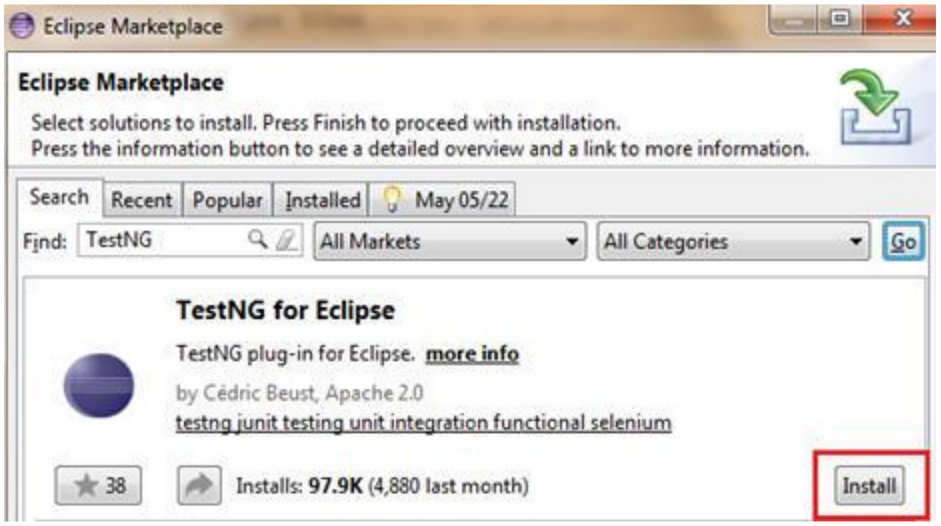
<u>Step 1</u>: Launch eclipse IDE -> Click on the Help option within the menu -> Select "Eclipse Marketplace.." option within the dropdown.



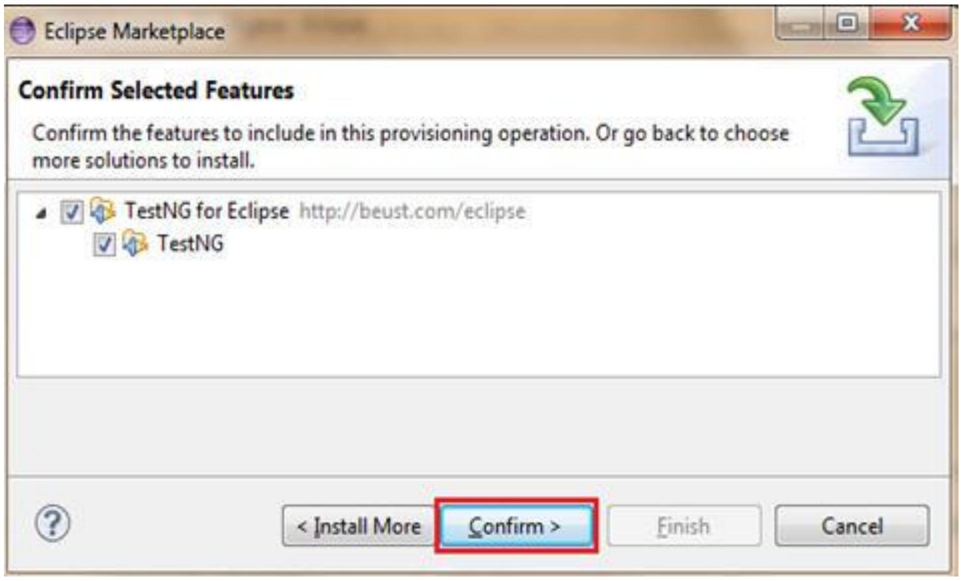
<u>Step 2</u>: Enter the keyword "TestNG" in the search textbox and click on "Go" button as shown below.


Eclipse Marketplace
Eclipse Marketplace
Select solutions to install. Press Finish to proceed with installation.
Press the information button to see a detailed overview and a link to more information.
Search
Recent
Popular
Installed
May 05/22
Find: TestNG
All Markets
All Categories
Go
Featured
Vaadin Plugin for Eclipse
Promoted - Vaadin is a powerful open source UI library for creating rich web user interfaces. It is a Java EE compatible component library that makes it easy... more info

Step 3: As soon as the user clicks on the "Go" button, the results matching to the search string would be displayed. Now user can click on the Install button to install TestNG.

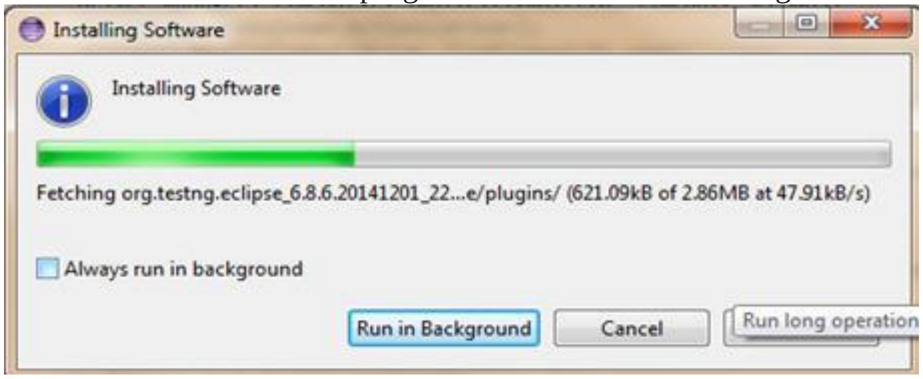

Eclipse Marketplace
Eclipse Marketplace
Select solutions to install. Press Finish to proceed with installation.
Press the information button to see a detailed overview and a link to more information.
Search
Recent
Popular
Installed
May 05/22
Find: TestNG
All Markets
All Categories
Go
TestNG for Eclipse
TestNG plug-in for Eclipse. more info
by Cédric Beust, Apache 2.0
testng junit testing unit integration functional selenium
38
Installs: 97.9K (4,880 last month)
Install

Step 4: As soon as the user clicks on the Install button, the user is prompted with a window to confirm the installation. Click on "Confirm" button.


Eclipse Marketplace
Confirm Selected Features
Confirm the features to include in this provisioning operation. Or go back to choose more solutions to install.
TestNG for Eclipse http://beust.com/eclipse
TestNG
< Install More
Confirm >
Finish
Cancel

<u>Step 5</u>: In the next step, the application would prompt you to accept the license and then click on the "Finish" button.

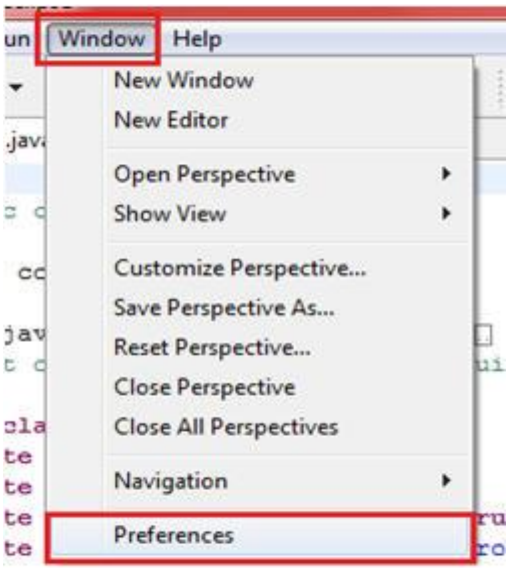
<u>Step 6</u>: The installation is initiated now and the progress can be seen as following:



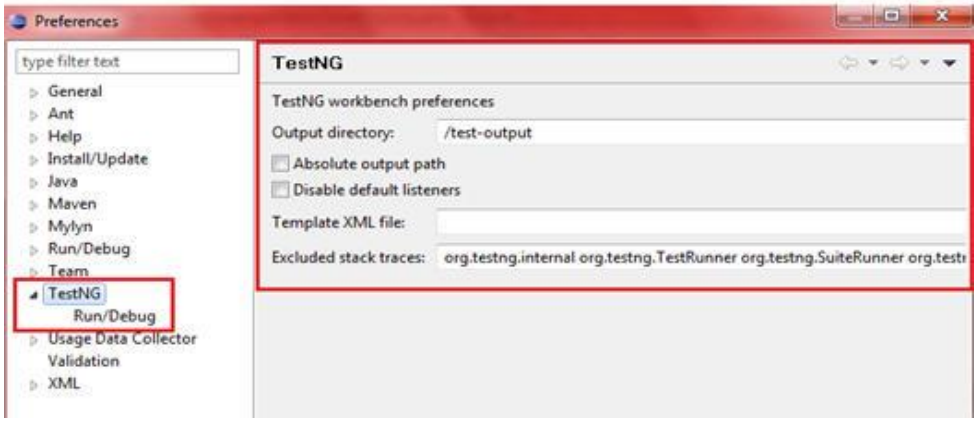
Fetching org.testng.eclipse_6.8.6.20141201_22...e/plugins/ (621.09kB of 2.86MB at 47.91kB/s)

We are advised to restart our eclipse so as to reflect the changes made.

Upon restart, user can verify the TestNG installation by navigating to "Preferences" from "Window" option in the menu bar. Refer the following figure for the same.



(Click on image to view enlarged)



TestNG

TestNG workbench preferences

Output directory: /test-output

Absolute output path

Disable default listeners

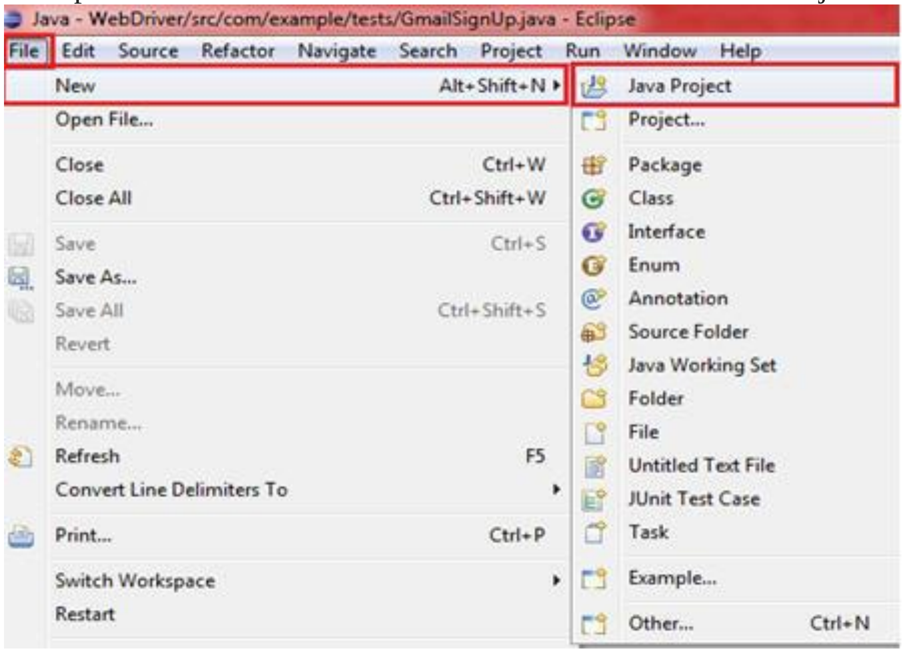
Template XML file:

Excluded stack traces: org.testng.internal org.testng.TestRunner org.testng.SuiteRunner org.testi

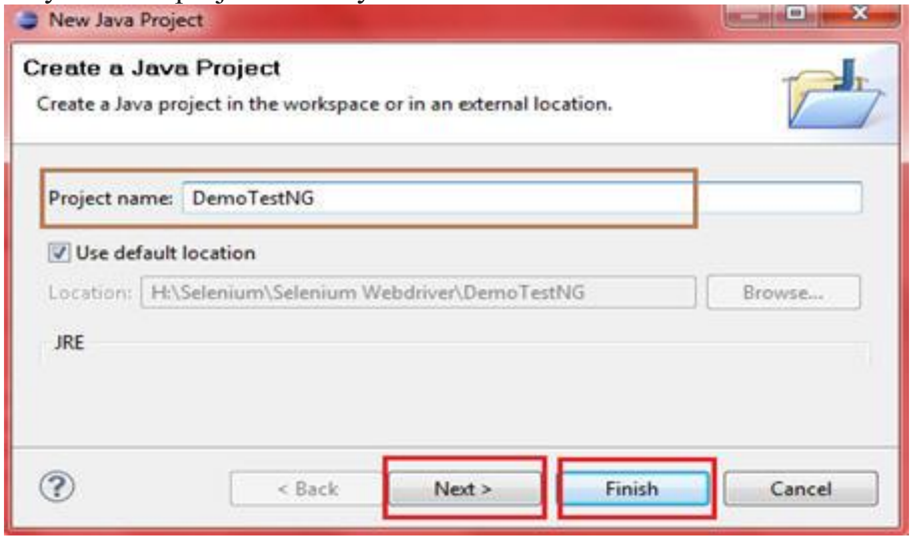
Creation of Sample TestNG project

Let us begin with the creation of TestNG project in eclipse IDE.

<u>Step 1</u>: Click on the File option within the menu -> Click on New -> Select Java Project.



<u>Step 2</u>: Enter the project name as "DemoTestNG" and click on "Next" button. As a concluding step, click on the "Finish" button and your Java project is ready.


New Java Project

Create a Java Project

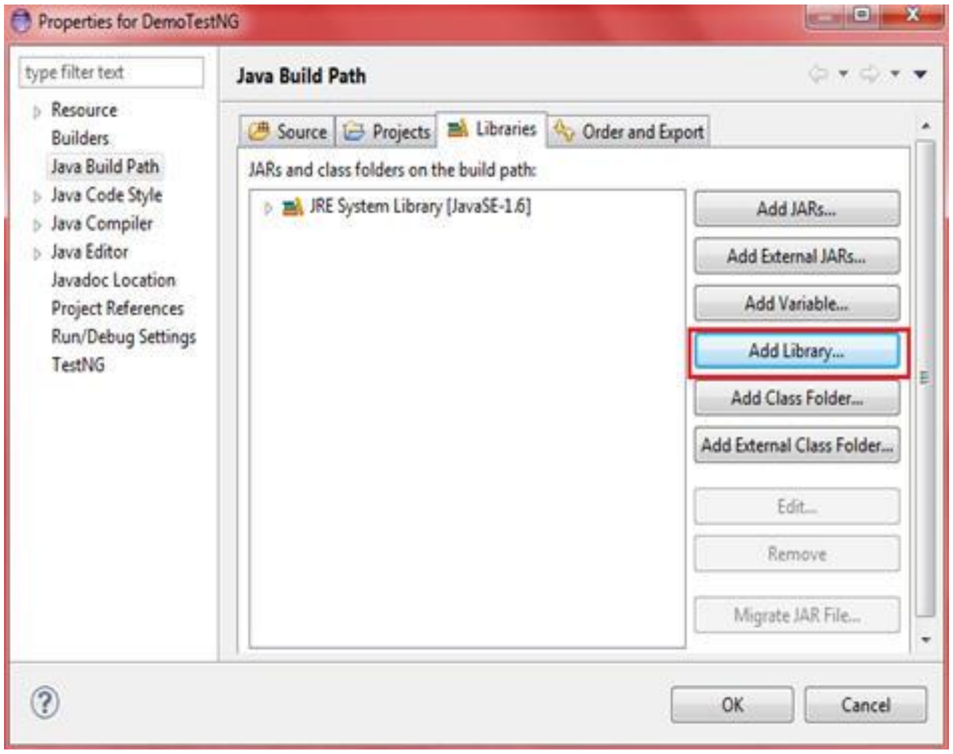
Create a Java project in the workspace or in an external location.

Project name: DemoTestNG

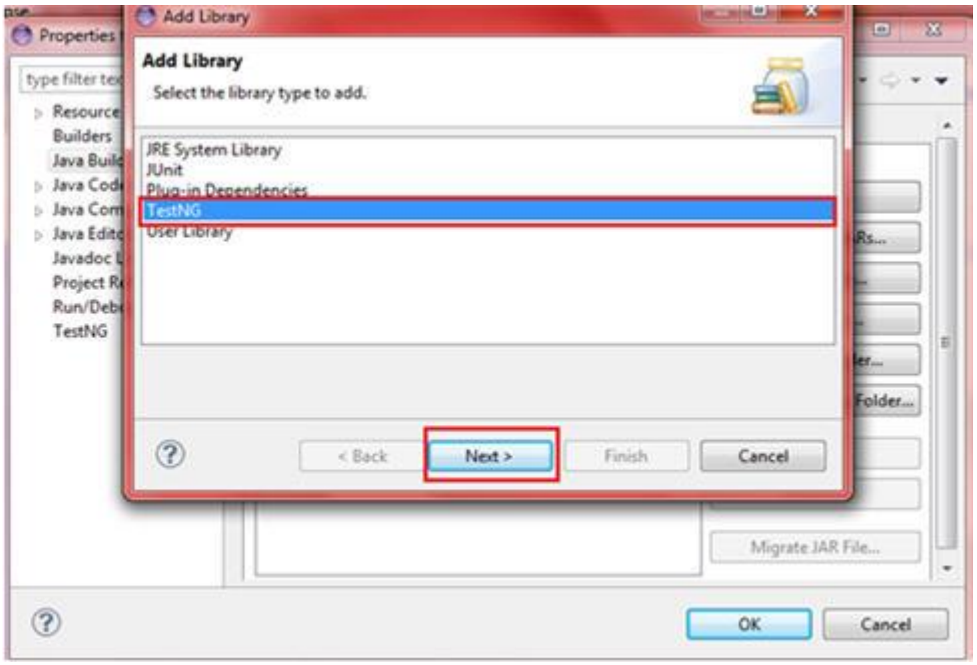
[✓] Use default location

Location: H:\Selenium\Selenium Webdriver\DemoTestNG

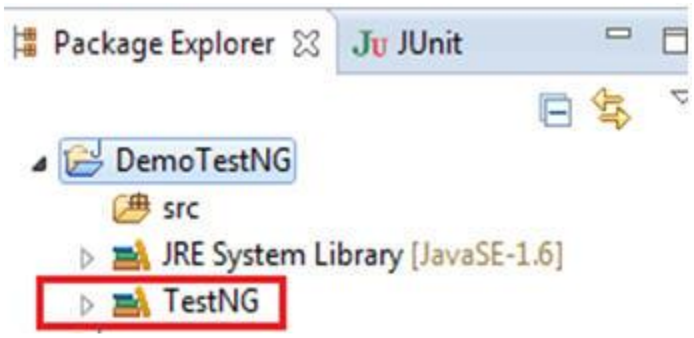
<u>Step 3</u>: The next step is to configure the TestNG library into the newly created Java project. For the same, Click on the "Libraries" tab under Configure Build Path. Click on "Add library" as shown below.



Step 4: The user would be subjected with a dialog box promoting him/her to select the library to be configured. Select TestNG and click on the "Next" button as shown below in the image. In the end, click on the "Finish" button.



The TestNG is now added to the Java project and the required libraries can be seen in the package explorer upon expanding the project.

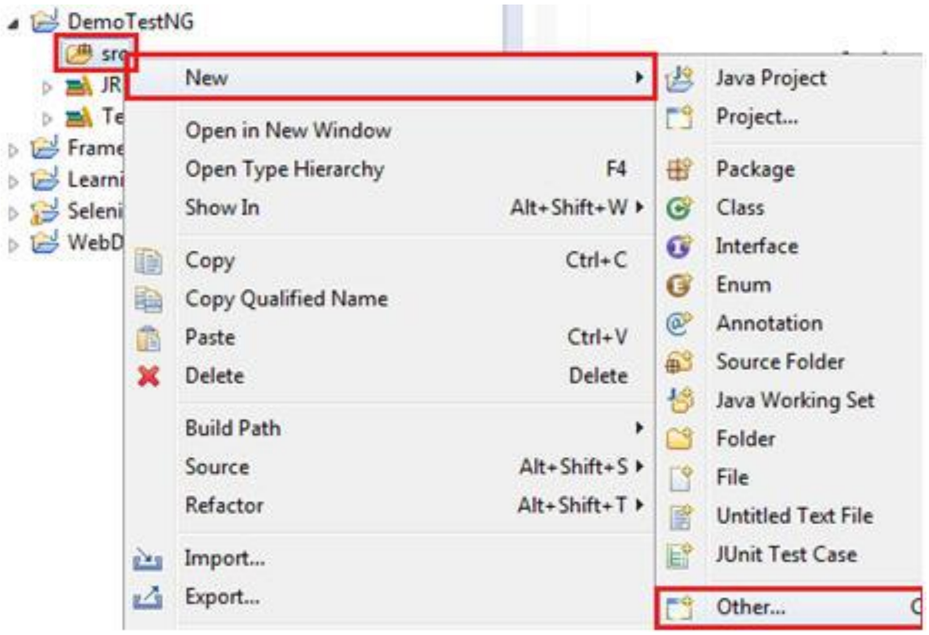


Add all the downloaded Selenium libraries and jars in the project's build path as illustrated in the previous tutorial.

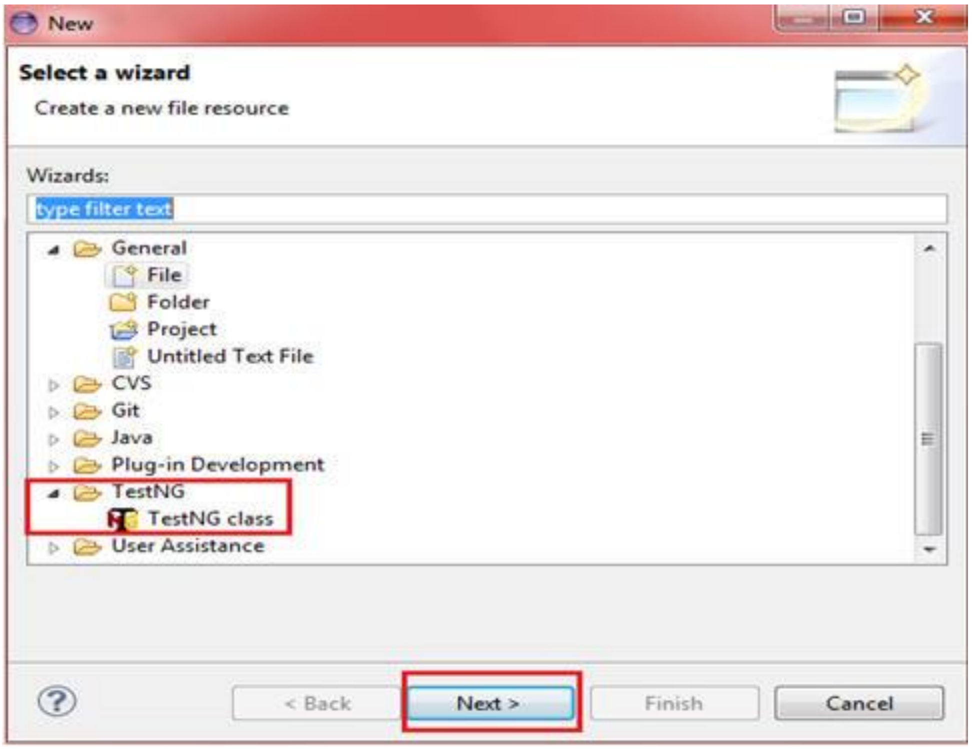
Creating TestNG class

Now that we have done all the basic setup to get started with the test script creation using TestNG. Let's create a sample script using TestNG.

<u>Step 1</u>: Expand the "DemoTestNG" project and traverse to "src" folder. Right click on the "src"package and navigate to New -> Other..



<u>Step 2</u>: Expand TestNG option and select "TestNG" class option and click on the "Next" button.


New

Select a wizard

Create a new file resource

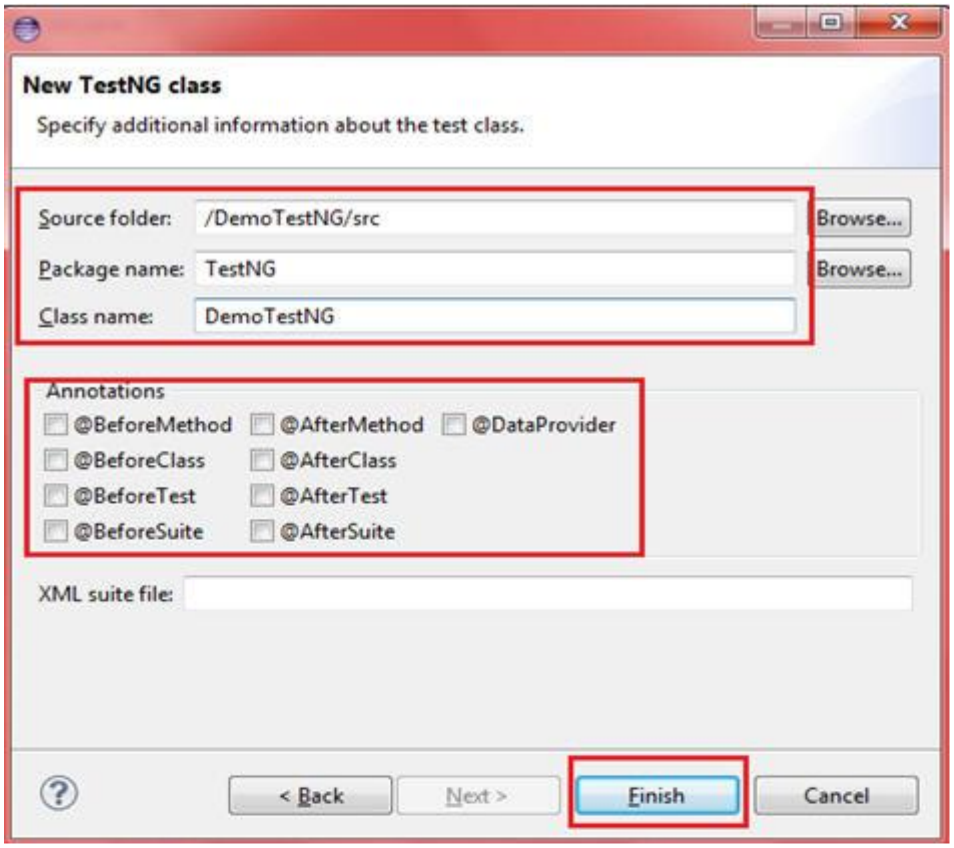
Wizards:

type filter text

- General
  - File
  - Folder
  - Project
  - Untitled Text File
- CVS
- Git
- Java
- Plug-in Development
- TestNG
  - TestNG class
- User Assistance

< Back | Next > | Finish | Cancel

<u>Step 3</u>: Furnish the required details as following. Specify the Source folder, package name and the TestNG class name and click on the Finish button. As it is evident from the below picture, user can also check various TestNG notations that would be reflected in the test class schema. TestNG annotations would be discussed later in this session.


New TestNG class

Specify additional information about the test class.

Source folder: /DemoTestNG/src

Package name: TestNG

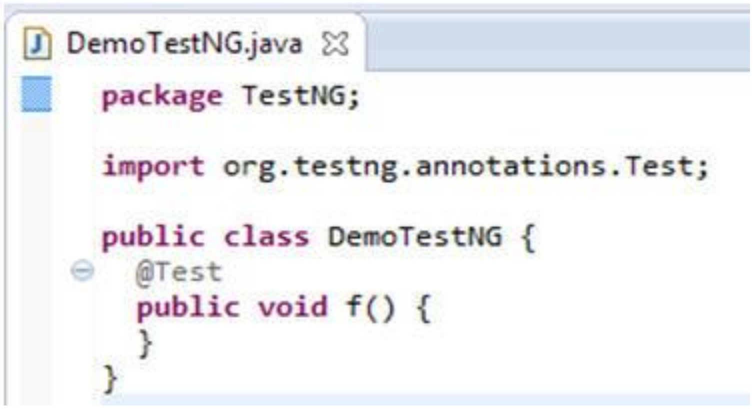
Class name: DemoTestNG

Annotations

- [ ] @BeforeMethod
- [ ] @AfterMethod
- [ ] @DataProvider
- [ ] @BeforeClass
- [ ] @AfterClass
- [ ] @BeforeTest
- [ ] @AfterTest
- [ ] @BeforeSuite
- [ ] @AfterSuite

XML suite file:

The above mentioned TestNG class would be created with the default schema.



```java
package TestNG;

import org.testng.annotations.Test;

public class DemoTestNG {
  @Test
  public void f() {
  }
}
```

Now that we have created the basic foundation for the TestNG test script, let us now inject the actual test code. We are using the same code we used in the previous session.

<u>Scenario</u>:

- Launch the browser and open “gmail.com”.
- Verify the title of the page and print the verification result.
- Enter the username and Password.
- Click on the Sign in button.
- Close the web browser.

<u>Code</u>:

```
package TestNG;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.testng.Assert;
import org.testng.annotations.Test;

public class DemoTestNG {
                      public WebDriver driver = new FirefoxDriver();
      String appUrl = &quot;https://accounts.google.com&quot;;

@Test
public void gmailLogin() {
           // launch the firefox browser and open the application url
            driver.get(&quot;https://gmail.com&quot;);

// maximize the browser window
            driver.manage().window().maximize();

// declare and initialize the variable to store the expected title of the webpage.
            String expectedTitle = &quot; Sign in - Google Accounts &quot;;

// fetch the title of the web page and save it into a string variable
            String actualTitle = driver.getTitle();
            Assert.assertEquals(expectedTitle,actualTitle);

// enter a valid username in the email textbox
            WebElement username = driver.findElement(By.id(&quot;Email&quot;));
            username.clear();
            username.sendKeys(&quot;TestSelenium&quot;);

// enter a valid password in the password textbox
            WebElement password = driver.findElement(By.id(&quot;Passwd&quot;));
            password.clear();
            password.sendKeys(&quot;password123&quot;);

// click on the Sign in button
            WebElement SignInButton = driver.findElement(By.id(&quot;signIn&quot;));
            SignInButton.click();

// close the web browser
            driver.close();
}
}
```

Code Explanation with respect to TestNG

1) @Test – @Test is one of the **TestNG annotations**. This annotation lets the program execution to know that method annotated as @Test is a test method. To be able to use different TestNG annotations, we need to import the package "**import** org.testng.annotations.*".

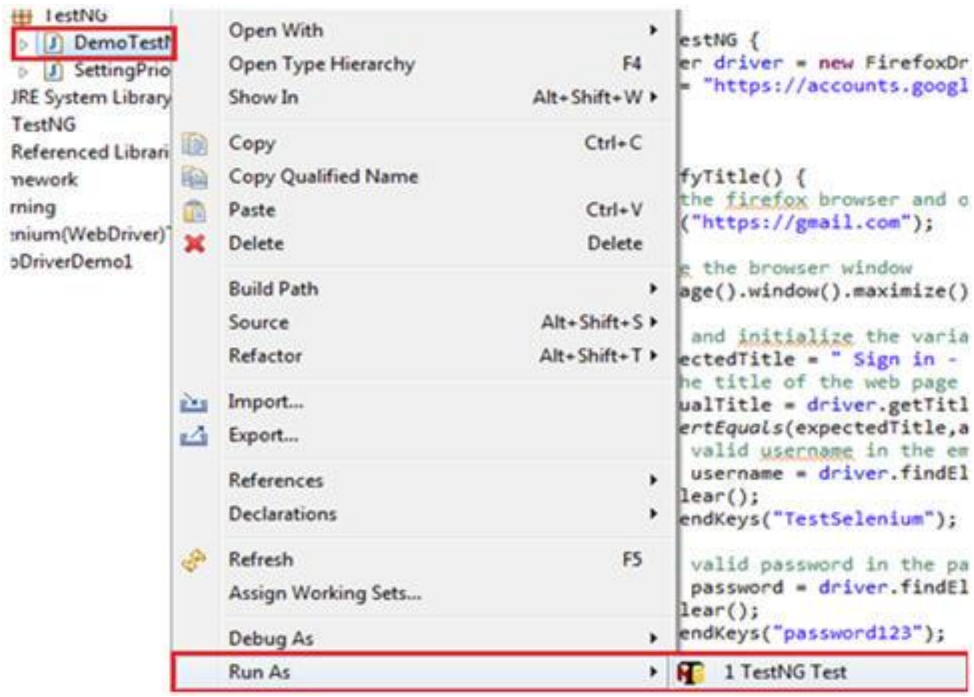
2) There is no need of main() method while creating test scripts using TestNG. The program execution is done on the basis of annotations.

3) In a statement, we used Assert class while comparing expected and the actual value. Assert class is used to perform various verifications. To be able to use different assertions, we are required to import "**import** org.testng.Assert".

Executing the TestNG script

The TestNG test script can be executed in the following way:

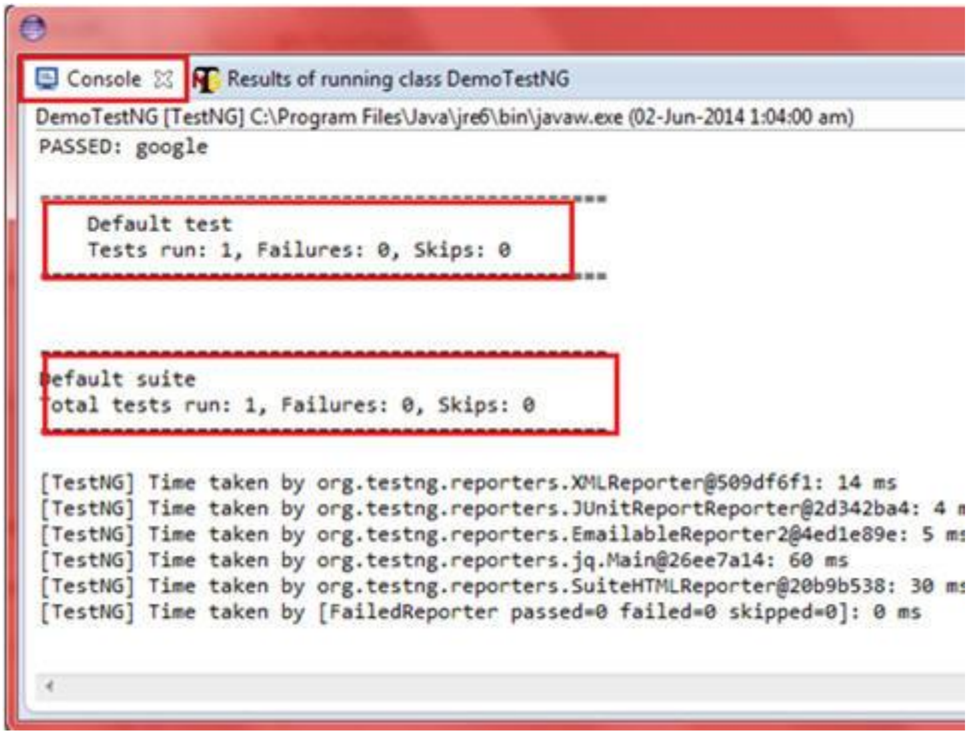
=> Right click anywhere inside the class within the editor or the java class within the package explorer, select "Run As" option and click on the "TestNG Test".



TestNG result is displayed into two windows:

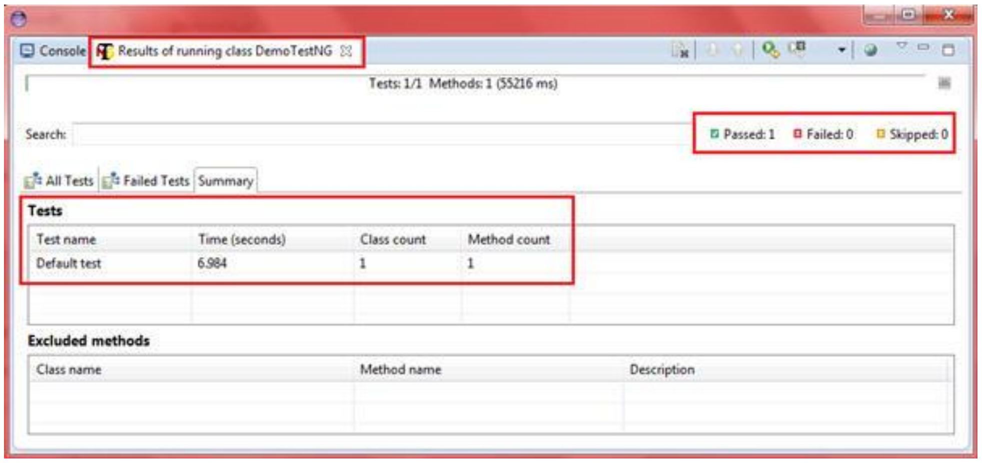
- Console Window
- TestNG Result Window

Refer the below screencasts for the result windows:



```
DemoTestNG [TestNG] C:\Program Files\Java\jre6\bin\javaw.exe (02-Jun-2014 1:04:00 am)
PASSED: google

===============================================
    Default test
    Tests run: 1, Failures: 0, Skips: 0
===============================================


===============================================
Default suite
Total tests run: 1, Failures: 0, Skips: 0
===============================================

[TestNG] Time taken by org.testng.reporters.XMLReporter@509df6f1: 14 ms
[TestNG] Time taken by org.testng.reporters.JUnitReportReporter@2d342ba4: 4 m
[TestNG] Time taken by org.testng.reporters.EmailableReporter2@4ed1e89e: 5 ms
[TestNG] Time taken by org.testng.reporters.jq.Main@26ee7a14: 60 ms
[TestNG] Time taken by org.testng.reporters.SuiteHTMLReporter@20b9b538: 30 ms
[TestNG] Time taken by [FailedReporter passed=0 failed=0 skipped=0]: 0 ms
```

(Click on image to view enlarged)



Console | Results of running class DemoTestNG

Tests: 1/1 Methods: 1 (55216 ms)

Search:

Passed: 1 Failed: 0 Skipped: 0

All Tests | Failed Tests | Summary

Tests

Test name	Time (seconds)	Class count	Method count
Default test	6.984	1	1

Excluded methods

Class name	Method name	Description

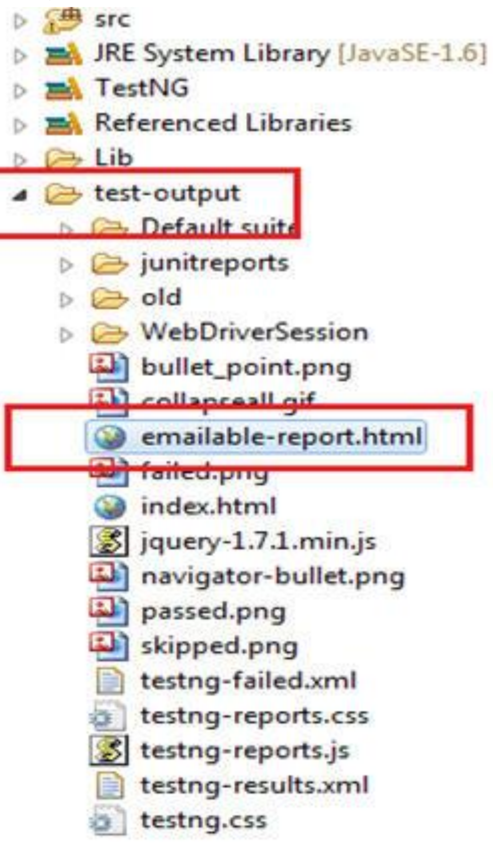
HTML Reports

TestNG comes with a great capability of generating user readable and comprehensible HTML reports for the test executions. These reports can be viewed in any of the browser and it can also be viewed using Eclipse's build –in browser support.

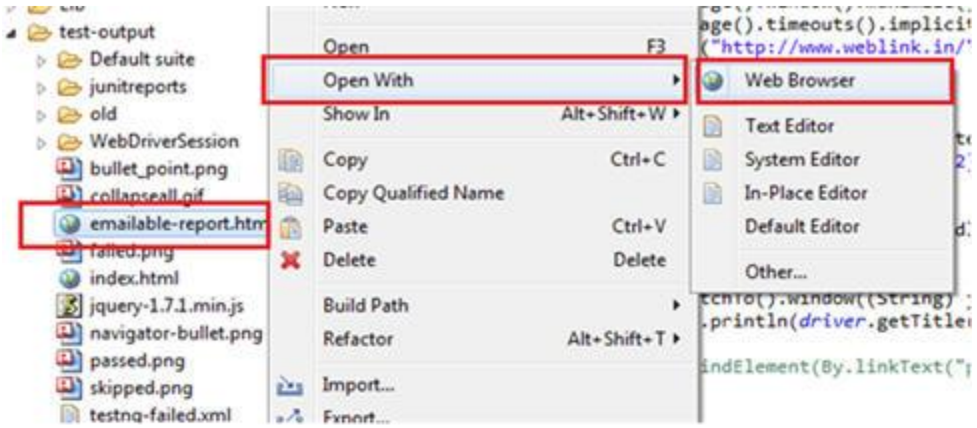
To generate the HTML report, follow the below steps:

Step 1: Execute the newly created TestNG class. Refresh the project containing the TestNG class by right clicking on it and selecting "Refresh" option.

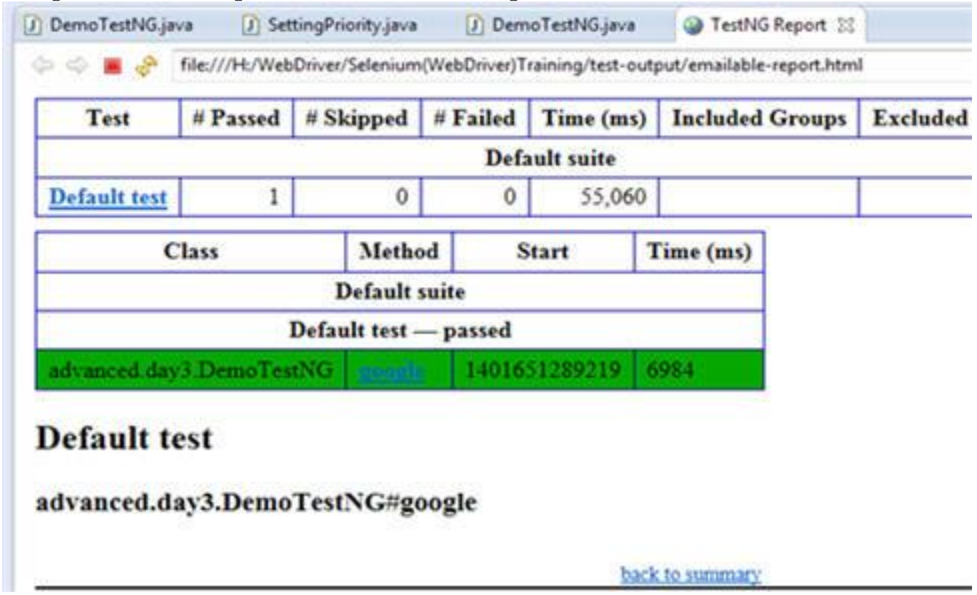
Step 2: A folder named as "test-output" shall be generated in the project at the "src" folder level. Expand the "test-output" folder and open on the "emailable-report.html" file with the Eclipse browser. The HTML file displays the result of the recent execution.



```
▷ src
▷ JRE System Library [JavaSE-1.6]
▷ TestNG
▷ Referenced Libraries
▷ Lib
◢ test-output
    ▷ Default suite
    ▷ junitreports
    ▷ old
    ▷ WebDriverSession
      bullet_point.png
      collapseall.gif
      emailable-report.html
      failed.png
      index.html
      jquery-1.7.1.min.js
      navigator-bullet.png
      passed.png
      skipped.png
      testng-failed.xml
      testng-reports.css
      testng-reports.js
      testng-results.xml
      testng.css
```



Step 3: The HTML report shall be opened with in the eclipse environment. Refer the below image for the same.



file:///H:/WebDriver/Selenium(WebDriver)Training/test-output/emailable-report.html

Test	# Passed	# Skipped	# Failed	Time (ms)	Included Groups	Excluded
Default suite						
Default test	1	0	0	55,060		

Class	Method	Start	Time (ms)
Default suite			
Default test — passed			
advanced.day3.DemoTestNG	google	1401651289219	6984

Default test

advanced.day3.DemoTestNG#google

back to summary

Refresh the page to see the results for fresh executions if any.

Setting Priority in TestNG

Code Snippet

```
package TestNG;
import org.testng.annotations.*;
public class SettingPriority {

@Test(priority=0)
public void method1() {
 }

@Test(priority=1)
            public void method2() {
 }

@Test(priority=2)
public void method3() {
 }
}
```

Code Walkthrough

If a test script is composed of more than one test method, the execution priority and sequence can be set using TestNG annotation “@Test” and by setting a value for the “priority” parameter.

In the above code snippet, all the methods are annotated with the help @Test and the priorities are set to 0, 1 and 2. Thus the order of execution in which the test methods would be executed is:

- Method1
- Method2
- Method3

Support for Annotations

There are number of annotations provided in TestNG and JUnit. The subtle difference is that TestNG provides some more advance annotations to JUnit.

TestNG Annotations:

Following is the list of the most useful and favorable annotations in TestNG:

Annotation	**Description**
@Test	The annotation notifies the system that the method annotated as @Test is a test method
@BeforeSuite	The annotation notifies the system that the method annotated as @BeforeSuite must be executed before executing the tests in the entire suite
@AfterSuite	The annotation notifies the system that the method annotated as @AfterSuite must be executed after executing the tests in the entire suite
@BeforeTest	The annotation notifies the system that the method annotated as @BeforeTest must be executed before executing any test method within the same test class
@AfterTest	The annotation notifies the system that the method annotated as @AfterTest must be executed after executing any test method within the same test class
@BeforeClass	The annotation notifies the system that the method annotated as @BeforeClass must be executed before executing the first test method within the same test class

Annotation	Description
@AfterClass	The annotation notifies the system that the method annotated as @AfterClass must be executed after executing the last test method within the same test class
@BeforeMethod	The annotation notifies the system that the method annotated as @BeforeMethod must be executed before executing any and every test method within the same test class
@AfterMethod	The annotation notifies the system that the method annotated as @AfterMethod must be executed after executing any and every test method within the same test class
@BeforeGroups	The annotation notifies the system that the method annotated as @BeforeGroups is a configuration method that enlists a group and that must be executed before executing the first test method of the group
@AfterGroups	The annotation notifies the system that the method annotated as @AfterGroups is a configuration method that enlists a group and that must be executed after executing the last test method of the group

Note: Many of the aforementioned annotations can be exercised in JUnit 3 and JUnit 4 framework also.

Conclusion

Through this tutorial, we tried to make you acquainted with a java based testing framework named as TestNG. We started off the session with the installation of the framework and moved with the script creation and advance topics. We discussed all the annotations provided by TestNG. We implemented and executed our first TestNG test script using annotations and assert statements.

Article summary:

- TestNG is an advance framework designed in a way to leverage the benefits by both the developers and testers.
- TestNG is an open source framework which is distributed under the Apache software License and is readily available for download.
- TestNG is considered to be superior to JUnit because of its advance features.
- **Features of TestNG**
  - Support for Annotations
  - Advance execution methodology that do not require test suites to be created

    - Support for parameterization
    - Support for Data Driven Testing using Dataproviders
    - Setting execution priorities for the test methods
    - Supports threat safe environment when executing multiple threads
    - Readily supports integration with various tools and plug-ins like build tools (Ant, Maven etc.), Integrated Development Environment (Eclipse).
    - Facilitates user with effective means of Report Generation using ReportNG
- **Advantages of TestNG over JUnit**
    - Added advance and easy annotations
    - Execution patterns can be set
    - Concurrent execution of test scripts
    - Test case dependencies can be set
- TestNG is freely available and can be easily installed in the Eclipse IDE using Eclipse Market.
- Upon installation, TestNG would be available as a library within the Eclipse environment.
- Create a new Java Project and configure the build path using TestNG library.
- Create a new TestNG class by expanding the created TestNG project and traverse to its “src” folder. Right click on the “src” package and navigate to New -> Other. Select TestNG class option.
- @Test is one of the annotations provided by TestNG. This annotation lets the program execution to know that method annotated as @Test is a test method. To be able to use different TestNG annotations, we need to import the package “**import** org.testng.annotations.*”.
- There is no need of main() method while creating test scripts using TestNG.
- We use Assert class while comparing expected and the actual value. Assert class is used to perform various verifications. To be able to use different assertions, we are required to import “**import** org.testng.Assert”.
- If a test script is composed of more than one test methods, the execution priority and sequence can be set using TestNG annotation “@Test” and by setting a value for the “priority” parameter.
- TestNG has a capability of generating human readable test execution reports automatically. These reports can be viewed in any of the browser and it can also be viewed using Eclipse’s built – in browser support.

Next Tutorial #13: Moving ahead with the upcoming tutorials in the Selenium series, we would concentrate on handling the various types of web elements available on the web pages. Therefore, **in the next tutorial, we would concentrate our focus on “dropdowns” and will exercise their handling strategies.** We would also discuss about WebDriver’s Select class and its methods to select values in the dropdowns.

A remark for the readers: While our next tutorial of the Selenium series is in the processing mode, readers can start creating their own basic WebDriver scripts using TestNG framework.

For more advance scripts and concepts, include as many annotations and assertions in your TestNG classes and execute them using TestNG environment. Also analyze the HTML reports generated by TestNG.

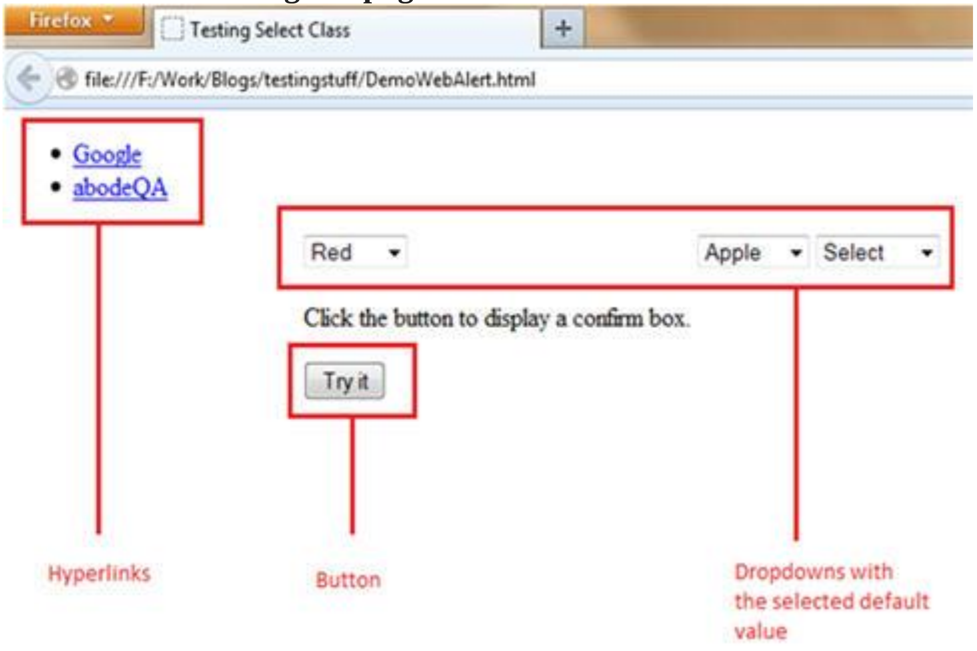
Usage of Selenium Select Class for Handling Dropdown Elements on a Web Page – Selenium Tutorial #13

In the previous tutorial, we studied about the various types of assert statements available in Java based unit testing framework and their applications with specimens. Re-iterating the fact that being an “Automation Test Engineer”, assertions play a very decisive and significant role in developing test scripts.

Moving ahead with the few upcoming tutorials in the Selenium series, we would concentrate on **handling the various types of web elements available on the web pages**. Therefore, in this tutorial, we would consider **“dropdowns” and exercise their handling strategies**.
Before moving towards problem statement and its resolution, let us take a moment to introduce and create an understanding regarding the application under test. As a sample, we have created a **dummy HTML page** consisting of multiple and assorted web elements.
The elementary web elements those constitute the web page are:

- Hyperlink
- Button
- Dropdown

Please take a reference of the following webpage aforementioned above:


Firefox
Testing Select Class
file:///F:/Work/Blogs/testingstuff/DemoWebAlert.html
Google
abodeQA
Red
Apple
Select
Click the button to display a confirm box.
Try it
Hyperlinks
Button
Dropdowns with the selected default value

Explanation of Application under Test

We have designed the web page in a way to include a few fundamental types of web elements.

- **Hyperlink**: The two hyperlinks namely “Google” and “abodeQA” have been provided that re-directs the user to “https://www.google.co.in/” and “http://www.abodeqa.com/” respectively on the click event.
- **Dropdown**: The three dropdowns have been created for selecting colors, fruits and animals with a value already set to default.
- **Button**: A “try it” button has been created to show up the pop up box having Ok and Cancel button upon click event.

Subsequent is the HTML code used to create the above mentioned webpage:

```
<!DOCTYPE html>
<html>
<head><title> Testing Select Class </title>
<body>
<div id="header">
<ul id="linkTabs">
<li>
<a href="https://www.google.co.in/">Google</a>
 </li>
<li>
<a href="http://abodeqa.wordpress.com/">abodeQA</a>
</li>
</ul>
</div>
<div class="header_spacer"></div>
<div id="container">
<div id="content" style="padding-left: 185px;">
<table id="selectTable">
<tbody>
<tr>
<td>
<div>
<select id="SelectID_One">
<option value="redvalue">Red</option>
<option value="greenvalue">Green</option>
<option value="yellowvalue">Yellow</option>
<option value="greyvalue">Grey</option>
</select>
</div>
</td>
<td>
<div>
<select id="SelectID_Two">
<option value="applevalue">Apple</option>
```

```
<option value="orangevalue">Orange</option>
<option value="mangovalue">Mango</option>
<option value="limevalue">Lime</option>
</select>
</div>
</td>
<td>
<div>
<select id="SelectID_Three">
<option value="selectValue">Select</option>
<option value="elephantvalue">Elephant</option>
<option value="mousevalue">Mouse</option>
<option value="dogvalue">Dog</option>
</select>
</div>
</td>
</tr>
<tr>
<td>
<!DOCTYPE html>
<html>
<body>
<p>Click the button to display a confirm box.</p>
<button onclick="myFunction()">Try it</button>
<script>
function myFunction()
{
confirm("Press a button!");
}
</script>
</body>
</html>
</td>
</tr>
</tbody>
</table>
</div>
</div>
</body>
</html>
```

Scenario to be automated

- Launch the web browser and open the webpage
- Click on the “Google” hyperlink
- Navigate back to the original web page

- Select the “Green” in color dropdown
- Select the “Orange” in the fruit dropdown
- Select the “Elephant” in the animal dropdown

WebDriver Code using Selenium Select Class

Please take a note that, for script creation, we would be using “Learning_Selenium” project created in the former tutorial.

Step 1: Create a new java class named as “HandlingDropDown” under the “Learning_Selenium” project.

Step 2: Copy and paste the below code in the “HandlingDropDown.java” class.

Below is the test script that is equivalent to the above mentioned scenario:

```
import static org.junit.Assert.*;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.support.ui.Select;

/**
 * class description
 */

public class HandlingDropDown {
    WebDriver driver;

    /**
     * Set up browser settings and open the application
     */

    @Before
    public void setUp() {
        driver=new FirefoxDriver();

// Opened the application
        driver.get("file:///F:/Work/Blogs/testingstuff/DemoWebAlert.html");
        driver.manage().window().maximize();
    }

    /**
     * Test to select the dropdown values
     * @throws InterruptedException
     */

    @Test
    public void testSelectFunctionality() throwsInterruptedException {

// Go to google
        driver.findElement(By.linkText("Google")).click();

// navigate back to previous webpage
        driver.navigate().back();
        Thread.sleep(5000);

// select the first operator using "select by value"
        Select selectByValue = newSelect(driver.findElement(By.id("SelectID_One")));
        selectByValue.selectByValue("greenvalue");
        Thread.sleep(5000);

// select the second dropdown using "select by visible text"
        Select selectByVisibleText = new Select (driver.findElement(By.id("SelectID_Two")));
        selectByVisibleText.selectByVisibleText("Lime");
        Thread.sleep(5000);

// select the third dropdown using "select by index"
        Select selectByIndex = newSelect(driver.findElement(By.id("SelectID_Three")));
        selectByIndex.selectByIndex(2);
        Thread.sleep(5000);
    }

    /**
     * Tear down the setup after test completes
     */

    @After
    public void tearDown() {
        driver.quit();
    }
}
```

Code Walkthrough

Import Statements

- ***import*** *org.openqa.selenium.support.ui.Select* – Import this package prior to the script creation. The package references to the Select class which is required to handle the dropdown.

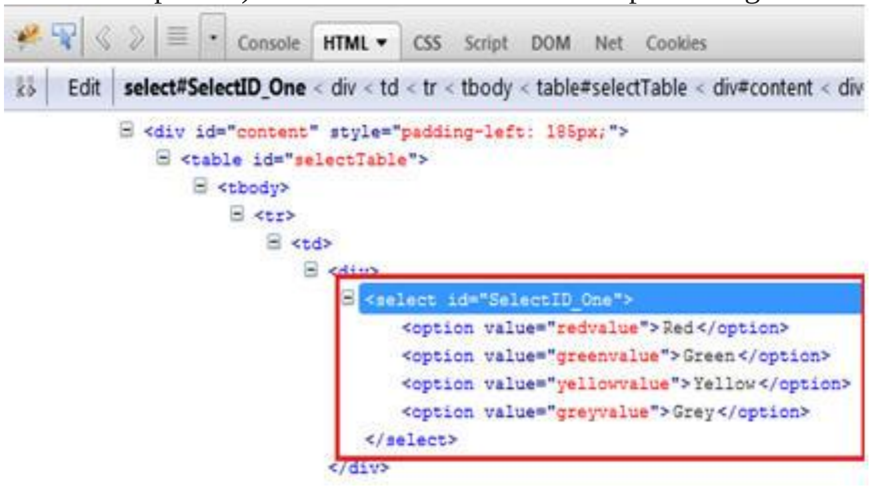
Object Instantiation for Select class

Select selectByValue = new Select(driver.findElement(By.id(“SelectID_One”)));

We create a reference variable for Select class and instantiate it using Select class and the identifier for the drop down.

The identifier or the locator value for the drop down can be found using the techniques discussed in the initial tutorials (by using Selenium IDE and firebug).

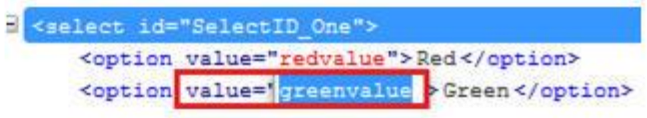
Take a notice that the identifier for a dropdown can be found as below:
<u>Step 1</u>: Most or almost all the dropdowns elements are defined in the <Select> tag having multiple values (values that can be set into the dropdown) that are defined under the <option> tags.



Console
HTML
CSS
Script
DOM
Net
Cookies
Edit
select#SelectID_One < div < td < tr < tbody < table#selectTable < div#content < div

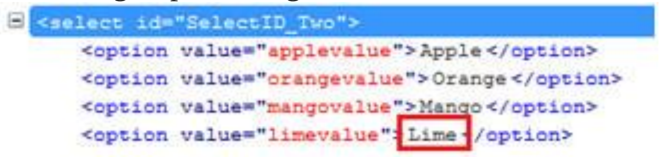
```
<div id="content" style="padding-left: 185px;">
  <table id="selectTable">
    <tbody>
      <tr>
        <td>
          <div>
            <select id="SelectID_One">
              <option value="redvalue">Red</option>
              <option value="greenvalue">Green</option>
              <option value="yellowvalue">Yellow</option>
              <option value="greyvalue">Grey</option>
            </select>
          </div>
```

Setting the value in the dropdown using *selectByValue()* method
selectByValue.selectByValue("greenvalue");
In the above java command, we select the value "green" in the drop down using the *selectByValue()* method and parameterizing it with the text present in the value attribute.



```
<select id="SelectID_One">
  <option value="redvalue">Red</option>
  <option value="greenvalue">Green</option>
```

Setting the value in the dropdown using selectByVisibleText() method
selectByValue.selectByVisibleText("Lime");
In the above java command, we select the value "Lime" in the drop down using
the *selectByVisibleText()* method and parameterizing it with the text present on the user interface or the text present between the opening and closing <option> tags.



```
<select id="SelectID_Two">
  <option value="applevalue">Apple</option>
  <option value="orangevalue">Orange</option>
  <option value="mangovalue">Mango</option>
  <option value="limevalue">Lime</option>
```

Setting the value in the dropdown using *selectByIndex()* method
selectByValue.selectByIndex("2");

In the above java command, we select the third value in the drop down using the *selectByIndex()* method and parameterizing it with the index value of the element which is desired to be selected in the dropdown.
Take a note that the index value starts with “0”.

Conclusion
In this tutorial, we tried to make you acquainted with the WebDriver’s Select class that is used to handle dropdown elements present on the web page. We also briefed you about the methods that can be used to populate the value in the dropdown.

Here is the article summary:

- WebDriver’s Select class is used to handle the dropdown elements present on a web page.
- Prior to the actual scripting, we need to import a package to be able to create a WebDriver script for handling a dropdown and making the Select class accessible.
  - ***import*** *org.openqa.selenium.support.ui.Select;*
- We create a reference variable for Select class and instantiate it using Select class and the identifier for the drop down.
  - Select *selectByValue = new Select(driver.findElement(By.id(“SelectID_One”)));*
- The identifier or the locator value for the drop can be found using Selenium IDE and firebug.
- Ideally there are three ways to select the desired value in the dropdown amongst the listed one.
  - *selectByValue()*
  - *selectByVisibleText()*
  - *selectByIndex()*
- The following java command is used to select the “green” color in the dropdown. Take a notice the value in the dropdown is selected using the *selectByValue()*
  - *selectByValue(“greenvalue”);*
- The following java command is used to select the “Lime” fruit in the dropdown. Take a notice the value in the dropdown is selected using the *selectByVisibleText()*
  - *selectByVisibleText(“Lime”);*
- The following java command is used to select the third value amongst all the available options enlisted for the dropdown. Take a notice the value in the dropdown is selected using the *selectByIndex()*
  - *selectByIndex(“2”);*

Next Tutorial #14: In the forthcoming tutorial, we would discuss about various types of commands in WebDriver like *isSelected(), isEnabled() and isDispalyed()* those return a Boolean value against the presence of a specified web element.
Till then, stay tuned and automate the dropdown using WebDriver utility – “Select class”.

Check Visibility of Web Elements Using Various Types WebDriver Commands – Selenium Tutorial #14

How to check visibility of web elements using various types of looping and conditional commands in WebDriver:

Previously in the series, we discussed about WebDriver's Select class which is primarily used to handle web elements like dropdowns and selecting various options under the dropdowns.

Moving ahead in the Selenium series, we would be discussing about the various types of looping and conditional commands in WebDriver like isSelected(), isEnabled() and isDispalyed(). These methods are used to determine the visibility scope for the web elements.

So let us start with a brief introduction – WebDriver has a W3C specification that details out the information about the different visibility preferences based out on the types of the web elements upon which the actions are to be performed.

WebDriver facilitates the user with the following methods to check the visibility of the web elements. These web elements can be buttons, dropboxes, checkboxes, radio buttons, labels etc.

- isDisplayed()
- isSelected()
- isEnabled()

For an improved understanding, let us discuss the aforementioned methods with code examples.

As a specimen, we would be using the "google.com" as an application under test and the "Learning_Selenium" project created in the previous tutorials for script generation.

Scenario to be automated

1. Launch the web browser and open the application under test – http://google.com
2. Verify the web page title
3. Verify if the "Google Search" button is displayed
4. Enter the keyword in the "Google Search" text box by which we would want to make the request
5. Verify that the "Search button" is displayed and enabled
6. Based on visibility of the Search button, click on the search button

WebDriver Code

Step 1: Create a new java class named as "VisibilityConditions" under the "Learning_Selenium" project.

Step 2: Copy and paste the below code in the "VisibilityConditions.java" class.

Below is the test script that is equivalent to the above mentioned scenario:

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;

public class VisibilityConditions {

    /**
     * @param args
     */

    public static void main(String[] args) {

        // objects and variables instantiation
        WebDriver driver = new FirefoxDriver();
        String appUrl = "https://google.com";

        // launch the firefox browser and open the application url
        driver.get(appUrl);

        // maximize the browser window
        driver.manage().window().maximize();

        // declare and initialize the variable to store the expected title of the webpage.
        String expectedTitle = "Google";

        // fetch the title of the web page and save it into a string variable
        String actualTitle = driver.getTitle();

        // compare the expected title of the page with the actual title of the page and print the result
        if (expectedTitle.equals(actualTitle))
        {
            System.out.println("Verification Successful - The correct title is displayed on the web page.");
        }
        else
        {
            System.out.println("Verification Failed - An incorrect title is displayed on the web page.");
        }

        // verify if the “Google Search” button is displayed and print the result
        booleansubmitbuttonPresence=driver.findElement(By.id("gbqfba")).isDisplayed();
         System.out.println(submitbuttonPresence);

        // enter the keyword in the “Google Search” text box by which we would want to make the request
        WebElement searchTextBox = driver.findElement(By.id("gbqfq"));
```

```
		searchTextBox.clear();
		searchTextBox.sendKeys("Selenium");

		// verify that the “Search button” is displayed and enabled
		boolean searchIconPresence = driver.findElement(By.id("gbqfb")).isDisplayed();
		boolean searchIconEnabled = driver.findElement(By.id("gbqfb")).isEnabled();

		if (searchIconPresence==true && searchIconEnabled==true)
		{
			// click on the search button
			WebElement searchIcon = driver.findElement(By.id("gbqfb"));
			searchIcon.click();
		}

		// close the web browser
		driver.close();
		System.out.println("Test script executed successfully.");

		// terminate the program
		System.exit(0);
	}
}
```

Code Walkthrough

Following are the ways in which we ascertain the presence of web elements on the web page.

boolean *submitbuttonPresence=driver.findElement(By.id(“gbqfba”)).isDisplayed();*
isDispalyed()

isDisplayed() is the method used to verify presence of a web element within the webpage. The method is designed to result a Boolean value with each success and failure. The method returns a “true” value if the specified web element is present on the web page and a “false” value if the web element is not present on the web page.

Thus the above code snippet verifies for the presence of submit button on the google web page and returns a true value if the submit button is present and visible else returns a false value if the submit button is not present on the web page.

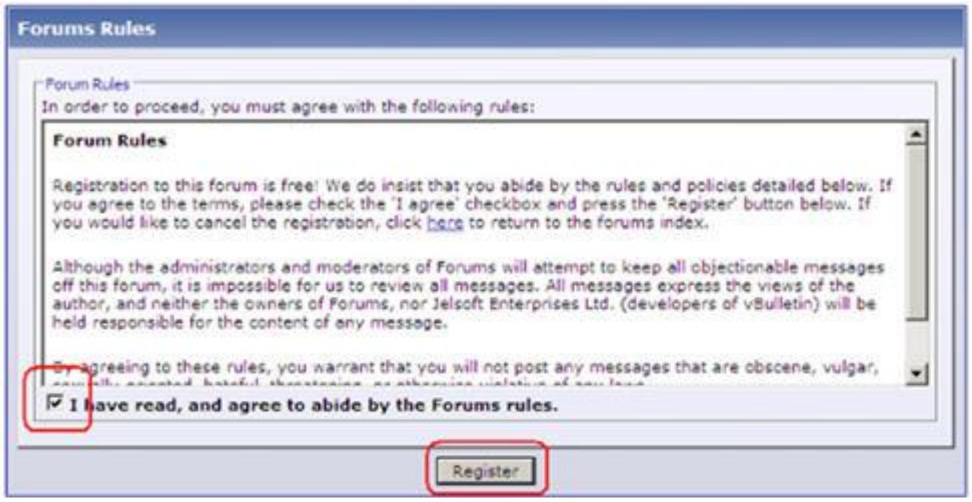
boolean *searchIconEnabled = driver.findElement(By.id(“gbqfb”)).isEnabled();*
The method deals with the visibility of all kinds of web elements not just limiting to any one type.

isEnabled()

isEnabled() is the method used to verify if the web element is enabled or disabled within the webpage. Like isDisplayed() method, it is designed to result a Boolean value with each success and failure. The method returns a “true” value if the specified web element is enabled on the web page and a “false” value if the web element is not enabled (state of being disabled) on the web page.

Thus the above code snippet verifies if the submit button is enabled or not and returns a Boolean value depending on the result.

The isEnabled() method is significant in scenarios where we want to ascertain that only if “Condition A” is fulfilled, then the element(principally button) is enabled. Refer the following illustration for the same.



Forums Rules

Forum Rules

In order to proceed, you must agree with the following rules:

Forum Rules

Registration to this forum is free! We do insist that you abide by the rules and policies detailed below. If you agree to the terms, please check the 'I agree' checkbox and press the 'Register' button below. If you would like to cancel the registration, click here to return to the forums index.

Although the administrators and moderators of Forums will attempt to keep all objectionable messages off this forum, it is impossible for us to review all messages. All messages express the views of the author, and neither the owners of Forums, nor Jelsoft Enterprises Ltd. (developers of vBulletin) will be held responsible for the content of any message.

By agreeing to these rules, you warrant that you will not post any messages that are obscene, vulgar,

☑ **I have read, and agree to abide by the Forums rules.**

Register

In the above figure, Register button button is enabled only when the agreement checkbox is selected.

Akin to above methods, we have a method referenced as “isSelected()” which tests if the specified web element is selected or not.

***boolean** searchIconSelected = driver.findElement(By.id(“male”)).isSelected();*
isSelected()
isSelected() is the method used to verify if the web element is selected or not. isSelected() method is predominantly used with radio buttons, dropdowns and checkboxes. Analogous to above methods, it is designed to result a Boolean value with each success and failure.

Thus the above code snippet verifies if the male radio button is selected or not and returns a Boolean value depending on the result. Refer the following image for the same.

Conclusion

In this tutorial, we tried to make you acquainted with the WebDriver's looping and conditional operations. These conditional methods often deal with almost all types of visibility options for web elements.

Article Summary:

- WebDriver has a W3C specification that details out the information about the different visibility preferences based out on the types of the web elements.
- isDisplayed() is the method used to verify presence of a web element within the webpage. The method returns a "true" value if the specified web element is present on the web page and a "false" value if the web element is not present on the web page.
- isDisplayed() is capable to check for the presence of all kinds of web elements available.
- isEnabled() is the method used to verify if the web element is enabled or disabled within the webpage.
- isEnabled() is primarily used with buttons.
- isSelected() is the method used to verify if the web element is selected or not. isSelected() method is pre-dominantly used with radio buttons, dropdowns and checkboxes.

Next Tutorial #15: While working on web applications, often we are re-directed to different web pages by refreshing the entire web page and re-loading the new web elements. At times there can be Ajax calls as well. Thus, a time lag can be seen while reloading the web pages and reflecting the web elements. **Thus, our next tutorial in-line is all about dealing with such time lags by using implicit and explicit waits.**

Note for the Readers: Till then, the reader can automate and test the visibility scope for the web elements using WebDriver's methods.

Practical Use of Different types of Selenium WebDriver Waits – Selenium Tutorial #15

In the previous tutorial, we tried to make you acquainted with the various WebDriver's looping and conditional operations. These conditional methods often deal with almost all types of visibility options for web elements.
Moving ahead in this free Selenium training series, we will discuss about **different types of waits provided by the WebDriver**. We will also discuss about v**arious types of navigation options** available in WebDriver.
Waits help the user to troubleshoot issues while re-directing to different web pages by refreshing the entire web page and re-loading the new web elements. At times there can be Ajax calls as well. Thus, a time lag can be seen while reloading the web pages and reflecting the web elements.
Users are often found navigating through various web pages back and forth. Thus, navigate() commands/methods provided by the WebDriver helps the user to simulate the real time scenarios by navigating between the web pages with reference to the web browser's history.

WebDriver equips the user with two genesis of waits in order to handle the recurring page loads, web element loads, appearance of windows, pop ups and error messages and reflection of web elements on the web page.

- Implicit Wait
- Explicit Wait

Let us discuss each of them in details considering practical approach.

WebDriver Implicit Wait
Implicit waits are used to provide a default waiting time (say 30 seconds) between each consecutive test step/command across the entire test script. Thus, subsequent test step would only execute when the 30 seconds have elapsed after executing the previous test step/command.

Key Notes

- Implicit wait is a single line of a code and can be declared in the setup method of the test script.
- When compared to Explicit wait, Implicit wait is transparent and uncomplicated. The syntax and approach is simpler than explicit wait.

Being easy and simple to apply, implicit wait introduces a few drawbacks as well. It gives rise to the test script execution time as each of the command would be ceased to wait for a stipulated amount of time before resuming the execution.

Thus, in order to trouble shoot this issue, WebDriver introduces Explicit waits where we can explicitly apply waits whenever the situation arises instead of forcefully waiting while executing each of the test step.

Import Statements

import *java.util.concurrent.TimeUnit* – To be able to access and apply implicit wait in our test scripts, we are bound to import this package into our test script.

Syntax

drv.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

Include the above line of code into your test script soon after instantiation of WebDriver instance variable. Thus, this is all what is required to set an implicit wait into your test script.

Code Walkthrough

The implicit wait mandates to pass two values as parameters. The first argument indicates the time in the numeric digits that the system needs to wait. The second argument indicates the time measurement scale. Thus, in the above code, we have mentioned the “30” seconds as default wait time and the time unit has been set to “seconds”.

WebDriver Explicit Wait

Explicit waits are used to halt the execution till the time a particular condition is met or the maximum time has elapsed. Unlike Implicit waits, Explicit waits are applied for a particular instance only.

WebDriver introduces classes like WebDriverWait and ExpectedConditions to enforce Explicit waits into the test scripts. In the ambit of this discussion, we will use “gmail.com” as a specimen.

Scenario to be automated

1. Launch the web browser and open the “gmail.com”
2. Enter a valid username
3. Enter a valid password
4. Click on the sign in button
5. Wait for Compose button to be visible after page load

WebDriver Code using Explicit wait

Please take a note that for script creation, we would be using “Learning_Selenium” project created in the former tutorials.

Step 1: Create a new java class named as “Wait_Demonstration” under the “Learning_Selenium” project.

Step 2: Copy and paste the below code in the “Wait_Demonstration.java” class.

Below is the test script that is equivalent to the above mentioned scenario.

```
import static org.junit.Assert.*;
import java.util.concurrent.TimeUnit;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
                                    import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;

public class Wait_Demonstration {

      // created reference variable for WebDriver
      WebDriver drv;
      @Before
      public void setup() throws InterruptedException {

            // initializing drv variable using FirefoxDriver
            drv=new FirefoxDriver();
            // launching gmail.com on the browser
            drv.get("https://gmail.com");
            // maximized the browser window
            drv.manage().window().maximize();
            drv.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
      }

      @Test
      public void test() throws InterruptedException {

            // saving the GUI element reference into a "username" variable of WebElement type
            WebElement username = drv.findElement(By.id("Email"));

            // entering username
            username.sendKeys("shruti.shrivastava.in");

            // entering password
            drv.findElement(By.id("Passwd")).sendKeys("password");

            // clicking signin button
            drv.findElement(By.id("signIn")).click();

            // explicit wait - to wait for the compose button to be click-able
            WebDriverWait wait = newWebDriverWait(drv,30);
```

```

        wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//div[contains(text(),'COMPOSE')]")));
            // click on the compose button as soon as the "compose" button is visible
       drv.findElement(By.xpath("//div[contains(text(),'COMPOSE')]")).click();
        }

       @After
       public void teardown() {
       // closes all the browser windows opened by web driver
   drv.quit();
       }
}
```

Import Statements

- ***import*** *org.openqa.selenium.support.ui.ExpectedConditions*
- ***import*** *org.openqa.selenium.support.ui.WebDriverWait*
- Import above packages prior to the script creation. The packages refer to the Select class which is required to handle the dropdown.

Object Instantiation for WebDriverWait class

WebDriverWait wait = ***new*** *WebDriverWait(drv,30);*

We create a reference variable "wait" for WebDriverWait class and instantiate it using WebDriver instance and maximum wait time for the execution to layoff. The maximum wait time quoted is measured in "seconds".

The WebDriver instantiation was discussed in the initial tutorials of WebDriver.

Expected Condition

wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//div[contains(text(),'COMPOSE')]")));
drv.findElement(By.xpath("//div[contains(text(),'COMPOSE')]")).click();

The above command waits for a stipulated amount of time or an expected condition to occur whichever occurs or elapses first.

Thus to be able to do this, we use the "wait" reference variable of WebDriverWait class created in the previous step with ExpectedConditions class and an actual condition which is expected to occur. Therefore, as soon as the expected condition occurs, the program control would move to the next execution step instead of forcefully waiting for the entire 30 seconds.

In our specimen, we wait for the "compose" button to be present and loaded as a part of home page load and thus, then we move forward with calling the click command on the "compose" button.

Types of Expected Conditions

ExpectedConditions class provides a great help to deal with scenarios where we have to ascertain for a condition to occur before executing the actual test step.

ExpectedConditions class comes with a wide range of expected conditions that can be accessed with the help of the WebDriverWait reference variable and until() method.

<u>Let us discuss a few of them at length:</u>

#1) elementToBeClickable() – The expected condition waits for an element to be clickable i.e. it should be present/displayed/visible on the screen as well as enabled.

Sample Code

wait.until(ExpectedConditions.elementToBeClickable(By.xpath("//div[contains(text(),'COMPOSE')]")));

#2) textToBePresentInElement() – The expected condition waits for an element having a certain string pattern.

Sample Code

wait.until(ExpectedConditions.textToBePresentInElement(By.xpath("//div[@id= 'forgotPass'"), "text to be found"));

#3) alertIsPresent()- The expected condition waits for an alert box to appear.

Sample Code

wait.until(ExpectedConditions.alertIsPresent()) !=null);

#4) titleIs() – The expected condition waits for a page with a specific title.

Sample Code

wait.until(ExpectedConditions.titleIs("gmail"));

#5) frameToBeAvailableAndSwitchToIt() – The expected condition waits for a frame to be available and then as soon as the frame is available, the control switches to it automatically.

Sample Code

wait.until(ExpectedConditions.frameToBeAvailableAndSwitchToIt(By.id("newframe")));

Navigation Using WebDriver

There is a very common user action where the user clicks on the back and forward buttons of the web browser back n forth to navigate to the different web pages visited in the current session on the browser's history. Thus to simulate such actions performed by the users, WebDriver introduces Navigate commands.

<u>Let us examine these commands in detail:</u>

#1) navigate().back()

This command lets the user to navigate to the previous web page.

Sample code:

driver.navigate().back();

The above command requires no parameters and takes back the user to the previous webpage in the web browser’s history.

#2) navigate().forward()

This command lets the user to navigate to the next web page with reference to the browser’s history.

Sample code:

driver.navigate().forward();

The above command requires no parameters and takes forward the user to the next webpage in the web browser’s history.

#3) navigate().refresh()

This command lets the user to refresh the current web page there by reloading all the web elements.

Sample code:

driver.navigate().refresh();

The above command requires no parameters and reloads the web page.

#4) navigate().to()

This command lets the user to launch a new web browser window and navigate to the specified URL.

Sample code:

driver.navigate().to(“http://google.com”);

The above command requires a web URL as a parameter and then it opens the specified URL on a freshly launched web browser.

Conclusion

In this tutorial, we tried to make you acquainted with the WebDriver’s waits. We discussed and exercised both the explicit and the implicit waits. At the same time, we also discussed about the different navigate commands.

Here are the cruxes of this article:

- WebDriver enables the user to choose amongst the available waits to handle situations where the execution flow may require a sleep for few seconds in order to load the web elements or to meet a specific condition. There are two types of waits available in WebDriver.
    - Implicit Wait
    - Explicit Wait
- **Implicit waits** are used to provide a default waiting time between each consecutive test step/command across the entire test script. Thus, subsequent test step would only execute when the specified amount of time have elapsed after executing the previous test step/command.
- **Explicit waits** are used to halt the execution till the time a particular condition is met or the maximum time has elapsed. Unlike Implicit waits, Explicit waits are applied for a particular instance only.
- WebDriver introduces classes like WebDriverWait and ExpectedConditions to enforce Explicit waits
- **ExpectedConditions** class provides a great help to deal with scenarios where we have to ascertain for a condition to occur before executing the actual test step.

- ExpectedConditions class comes with a wide range of expected conditions that can be accessed with the help of the WebDriverWait reference variable and until() method.
- **Navigate() methods**/commands are used to simulate the user behavior while navigating between various web pages back and forth.

Next Tutorial #16: Coming on to the next tutorial in the list, we would make the users familiar with various types of alerts that may appear while accessing web sites and their handling approaches in WebDriver. The types of alerts that we would be focusing on are majorly – windows based alert pop ups and web based alert pop ups. As we know that handling windows based pop ups is beyond WebDriver's capabilities, thus we would also exercise some third party utilities to handle window pop ups.

Note for the Readers: Till then, the readers can automate the scenarios having various page loads and dynamic elements popping up on to the screen using the various expected conditions and navigate commands.

How to Handle Alerts/Popups in Selenium WebDriver – Selenium Tutorial #16

Efficient Ways to Handle Windows and Web based Alerts/Popups in Selenium WebDriver:
In the previous tutorial, we focused our discussion on different types of waits provided by the WebDriver. We also discussed about various types of navigation options available in WebDriver.
Moving ahead in the Selenium WebDriver Tutorials, we will discuss about **different types of alerts available while testing web applications and their handling strategies.**

There are two types of alerts that we would be focusing on majorly:

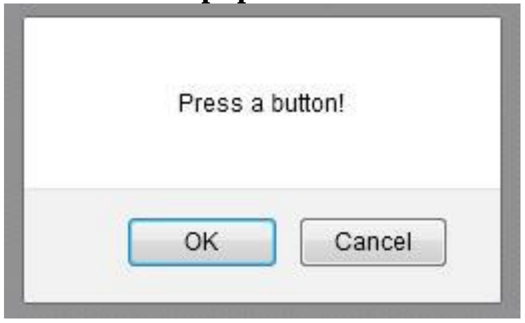
1. Windows based alert pop ups
2. Web based alert pop ups

As we know that handling windows based pop ups is beyond WebDriver's capabilities, thus we would exercise some third party utilities to handle window pop ups.
Handling pop up is one of the most challenging piece of work to automate while testing web applications. Owing to the diversity in types of pop ups complexes the situation even more.

What is Alert box/ Pop up box/ confirmation Box/ Prompt/ Authentication Box?
It is nothing but a small box that appears on the display screen to give you some kind of information or to warn you about a potentially damaging operation or it may even ask you for the permissions for the operation.

Example: Let us consider a real life example for a better understanding; Let us assume that we uploaded a photograph on any of these popular social networking sites. Later on, i wish to delete the uploaded photograph. So in order to delete, i clicked on the delete button. As soon as I click on the delete button, the system warns me against my action, prompting – Do you really want to delete the file? So now we have an option to either accept this alert or reject it.
So ahead in the session, **let's see how do we reject or accept the alerts depending on their types.** Starting with the web based pop ups.

Web Based Popups


Press a button!

Let us see how do we handle them using WebDriver.

Handling web based pop-up box

WebDriver offers the users with a very efficient way to handle these pop ups using Alert interface.

There are the four methods that we would be using along with the Alert interface.

1) ***void dismiss()*** – The dismiss() method clicks on the “Cancel” button as soon as the pop up window appears.

2) ***void accept()*** – The accept() method clicks on the “Ok” button as soon as the pop up window appears.

3) ***String getText()*** – The getText() method returns the text displayed on the alert box.

4) ***void sendKeys(String stringToSend)*** – The sendKeys() method enters the specified string pattern into the alert box.

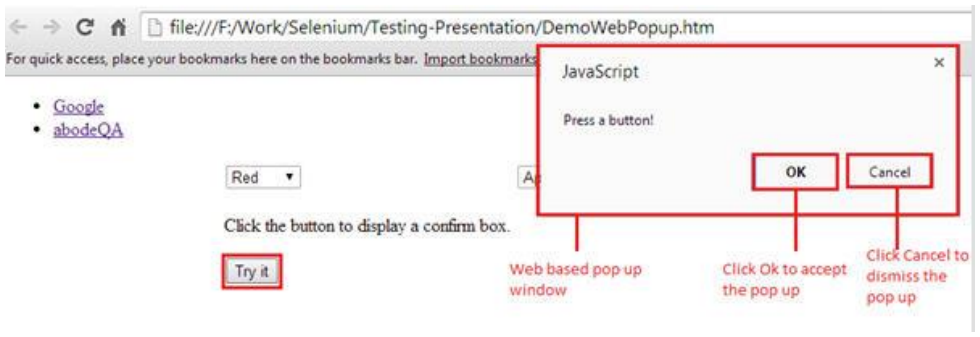
Let us move ahead and look at the actual implementation.

<u>Explanation of Application under Test</u>

We have designed a web page in a way to include a few fundamental types of web elements. This is the same application we introduced while discussing Select class earlier in this series.

- **Hyperlink**: The two hyperlinks namely “Google” and “abodeQA” have been provided that re-directs the user to “http://www.google.com/” and “http://www.abodeqa.com/” respectively on the click event.
- **Dropdown**: The three hyperlinks have been created for selecting colors, fruits and animals with a value set to default.
- **Button**: A “try it” button has been created to show up the pop up box having OK and Cancel buttons upon click event.

(Click on image to view enlarged)


file:///F:/Work/Selenium/Testing-Presentation/DemoWebPopup.htm
For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks
Google
abodeQA
Red
Click the button to display a confirm box.
Try it
JavaScript
Press a button!
OK
Cancel
Web based pop up window
Click Ok to accept the pop up
Click Cancel to dismiss the pop up

Subsequent is the HTML code used to create the above mentioned webpage:

```
<!DOCTYPE html></pre>
<html>
<head><title> Testing Select Class </title>
<body>
<div id="header">
```

```
<ul id="linkTabs">
<li>
<a href="https://www.google.com/">Google</a>
 </li>
<li>
<a href="http://abodeqa.wordpress.com/">abodeQA</a>
</li>
</ul>
</div>
<div class="header_spacer"></div>
<div id="container">
<div id="content" style="padding-left: 185px;">
<table id="selectTable">
<tbody>
<tr>
<td>
<div>
<select id="SelectID_One">
<option value="redvalue">Red</option>
<option value="greenvalue">Green</option>
<option value="yellowvalue">Yellow</option>
<option value="greyvalue">Grey</option>
</select>
</div>
</td>
<td>
<div>
<select id="SelectID_Two">
<option value="applevalue">Apple</option>
<option value="orangevalue">Orange</option>
<option value="mangovalue">Mango</option
>
<option value="limevalue">Lime</option>
</select>
</div>
</td>
<td>
<div>
<select id="SelectID_Three">
<option value="selectValue">Select</option>
<option value="elephantvalue">Elephant</option>
<option value="mousevalue">Mouse</option>
<option value="dogvalue">Dog</option>
</select>
</div>
```

```
</td>
</tr>
<tr>
<td>

<!DOCTYPE html>
<html>
<body>
<p>Click the button to display a confirm box.</p>
<button onclick="myFunction()">Try it</button>

<script>
function myFunction()
{
confirm("Press a button!");
}
</script>
</body>
</html>
</td>
</tr>
</tbody>
</table>
</div>
</div>
</body>
</html>
```

Scenario to be automated

1. Launch the web browser and open the webpage
2. Click on the “Try it” button
3. Accept the alert
4. Click on the “Try it” button again
5. Reject the alert

WebDriver Code using Select Class

Please take a note that for script creation, we would be using “Learning_Selenium” project created in the former tutorial.

Step 1: Create a new java class named as “DemoWebAlert” under the “Learning_Selenium” project.
Step 2: Copy and paste the below code in the “DemoWebAlert.java” class.
Below is the test script that is equivalent to the above mentioned scenario.

```
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import org.openqa.selenium.Alert;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

/**
 * class description
*/

public class DemoWebAlert
{
        WebDriver driver;
        /**
        * Constructor
        */
        public DemoWebAlert() {
        }

        /**
        * Set up browser settings and open the application
        */

        @Before
        public void setUp() {
                driver=newFirefoxDriver();
                // Opened the application
                driver.get("file:///F:/Work/Selenium/Testing-Presentation/DemoWebPopup.htm");
                driver.manage().window().maximize();
        }

        /**
        * Test to check Select functionality
        * @throws InterruptedException
        */

        @Test
        public void testWebAlert() throwsInterruptedException {
                // clicking on try it button
                driver.findElement(By.xpath("//button[contains(text(),'Try it')]")).click();
                 Thread.sleep(5000);

                // accepting javascript alert
                Alert alert = driver.switchTo().alert();
```

```
                    alert.accept();

                    // clicking on try it button
                    driver.findElement(By.xpath("//button[contains(text(),'Try it')]")).click();
                     Thread.sleep(5000);

                    // accepting javascript alert
                    driver.switchTo().alert().dismiss();

                     // clicking on try it button
                    driver.findElement(By.xpath("//button[contains(text(),'Try it')]")).click();
                    Thread.sleep(5000);

                     // accepting javascript alert
                    System.out.println(driver.switchTo().alert().getText());
                    driver.switchTo().alert().accept();
            }

            /**
            * Tear down the setup after test completes
            */

            @After
            public void tearDown() {
              driver.quit();
            }
}
```

Code Walk-through

Import Statements

Import org.openqa.selenium.Alert – Import this package prior to the script creation The package references to the Alert class which is required to handle the web based alerts in WebDriver.

Object Creation for Alert class

Alert alert = driver.switchTo().alert();

We create a reference variable for Alert class and references it to the alert.

Switch to Alert

Driver.switchTo().alert();

The above command is used to switch the control to the recently generated pop up window.

Accept the Alert

alert.accept();

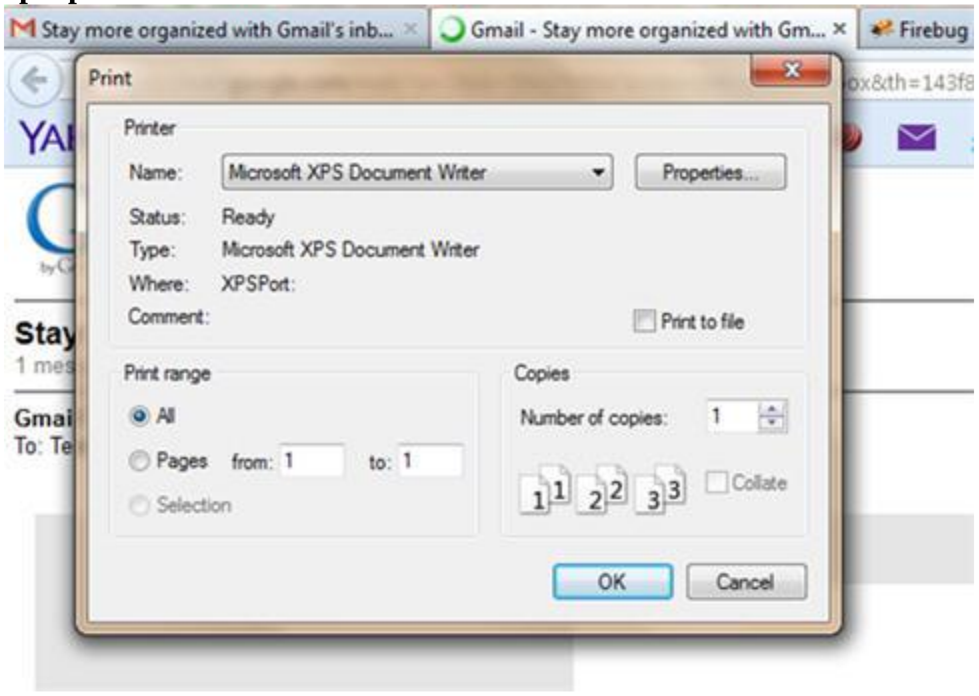
The above command accepts the alert thereby clicking on the Ok button.

Reject the Alert

alert.dismiss();

The above command closes the alert thereby clicking on the Cancel button and hence the operation should not proceed.

Window Based Pop Ups



At times while automating, we get some scenarios, where we need to handle pop ups generated by windows like a print pop up or a browsing window while uploading a file.

Handling these pop-ups have always been a little tricky as we know Selenium is an automation testing tool which supports only web application testing, that means, it doesn't support windows based applications and window alert is one of them. However Selenium alone can't help the situation but along with some third party intervention, this problem can be overcome.

There are several third party tools available for handling window based pop-ups along with the selenium.

So now let's handle a window based pop up using Robot class.

Robot class is a java based utility which emulates the keyboard and mouse actions.

Before moving ahead, let us take a moment to have a look at the application under test (AUT).

<u>**Explanation of Application under Test**</u>

As an application under test, we would be using "gmail.com". I believe the application doesn't require any more introductions.

Scenario to be automated

1. Launch the web browser and open the application – “gmail.com”
2. Enter valid username and password
3. Click on the sign in button
4. Click on the a compose button
5. Click on the attach icon
6. Select the files to be uploaded with the window based pop up.

WebDriver Code using Robot Class

Please take a note that for script creation, we would be using “Learning_Selenium” project created in the former tutorial.

Step 1: Create a new java class named as “DemoWindowAlert” under the “Learning_Selenium” project.

Step 2: Copy and paste the below code in the “DemoWindowAlert.java” class.

Below is the test script that is equivalent to the above mentioned scenario.

```
import java.awt.Robot;</pre>
import java.awt.event.KeyEvent;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

public class DemoWindowAlert {
WebDriver driver;
@Before

public void setUp()
{
driver=new FirefoxDriver();
driver.get("https://gmail.com");
driver.manage().window().maximize();
}

@Test
public void testWindowAlert() throws Exception{

// enter a valid email address
driver.findElement(By.id("Email")).sendKeys("TestSelenium1607@gmail.com");

// enter a valid password
driver.findElement(By.id("Passwd")).sendKeys("TestSelenium");

// click on sign in button
driver.findElement(By.id("signIn")).click();
Thread.sleep(30000);

// click on compose button
driver.findElement(By.xpath("//div[@class='z0']//div[contains(text(),'COMPOSE')]")).click();

 // click on attach files icon
driver.findElement(By.xpath("//div[contains(@command,'Files')]//div[contains(@class,'aaA')]")).click();

// creating instance of Robot class (A java based utility)
Robot rb =new Robot();

// pressing keys with the help of keyPress and keyRelease events
rb.keyPress(KeyEvent.VK_D);
rb.keyRelease(KeyEvent.VK_D);
Thread.sleep(2000);

rb.keyPress(KeyEvent.VK_SHIFT);
rb.keyPress(KeyEvent.VK_SEMICOLON);
rb.keyRelease(KeyEvent.VK_SEMICOLON);
rb.keyRelease(KeyEvent.VK_SHIFT);

rb.keyPress(KeyEvent.VK_BACK_SLASH);
rb.keyRelease(KeyEvent.VK_BACK_SLASH);
Thread.sleep(2000);

rb.keyPress(KeyEvent.VK_P);
rb.keyRelease(KeyEvent.VK_P);

rb.keyPress(KeyEvent.VK_I);
rb.keyRelease(KeyEvent.VK_I);

rb.keyPress(KeyEvent.VK_C);
rb.keyRelease(KeyEvent.VK_C);
Thread.sleep(2000);

rb.keyPress(KeyEvent.VK_ENTER);
rb.keyRelease(KeyEvent.VK_ENTER);
Thread.sleep(2000);
}

@After
```

```
public void tearDown()
{
driver.quit();
}
}
```

Code Walk-through

Import Statements

import java.awt.Robot – Import this package prior to the script creation The package references to the Robot class in java which is required simulate keyboard and mouse events.

import java.awt.event.KeyEvent – The package allows the user to use keyPress and keyRelease events of keyboard.

Object Creation for Robot class

Robot rb =new Robot();

We create a reference variable for Robot class and instantiate it.

KeyPress and KeyRelease Events

rb.keyPress(KeyEvent.VK_D);

rb.keyRelease(KeyEvent.VK_D);

The keyPress and keyRelease methods simulate the user pressing and releasing a certain key on the keyboard respectively.

Conclusion

In this tutorial, we tried to make you acquainted with the WebDriver's Alert class that is used to handle web based pop ups. We also briefed you about the Robot class that can be used to populate the value in the window based alert with the help of keyPress and keyRelease events.

Article summary:

- Alerts are a small box that appears on the display screen to give you some kind of information or to warn you about a potentially damaging operation or it may even ask you for the permissions for the operation.
- **There are popularly two types of alerts**–
  - Windows based alert pop ups
  - Web based alert pop ups
- Prior to the actual scripting, we need to import a package to be able to create a WebDriver script for handling a dropdown and making the Select class accessible.
- WebDriver offers the users with a very efficient way to handle these pop ups using Alert interface.
- *void dismiss()* – The *dismiss()* method clicks on the "Cancel" button as soon as the pop up window appears.
- *void accept()* – The *accept()* method clicks on the "Ok" button as soon as the pop up window appears.

- String *getText()* – The *getText()* method returns the text displayed on the alert box.
- *void sendKeys(String stringToSend)* – The *sendKeys()* method enters the specified string pattern into the alert box.
- **Handling window based pop-ups** have always been a little tricky as we know Selenium is an automation testing tool which supports only web application testing, that means, it doesn't support windows based applications and window alert is one of them.
- **Robot class** is a java based utility which emulates the keyboard and mouse actions and can be effectively used to handling window based pop up with the help of keyboard events.
- The keyPress and keyRelease methods simulate the user pressing and releasing a certain key on the keyboard respectively.

Next Tutorial #17: In the upcoming tutorial, we would discuss about the various other **commonly used WebDriver commands**. We would shed light on topics like exception handling and iframe handling. We would also discuss about the get commands provided in WebDriver.
We would explain these topics with quick examples in order to make them understandable for the readers to exercise these concepts in their day to day scripting.

Note for the Readers: Till then, stay tuned and automate the web pages having web based and window based pop ups using WebDriver utility – "Alert class" and Java utility – "Robot Class".
Feel free to post your queries/comments about this or any other previous tutorials in comments below.

Various Commonly and Routinely Used Selenium WebDriver Commands – Selenium Tutorial #17

In the last tutorial, we discussed about the different types of alertsencountered while testing web based applications and their effective ways of handling. We discussed both the types of alerts i.e. "Web-based alerts" and "Window-based alerts" at length. We also made you acquainted with yet another Java based utility named as "Robot Class" to handle windows-based pop up.
Advancing ahead in this Selenium WebDriver tutorial series, we would be pressing on **various commonly and routinely used Selenium WebDriver commands**. We will precisely and briefly discuss each of these Selenium commands so as to make you capable of using these commands effectively whenever the situation arises. **Selenium WebDriver Commands**:
Just to have a rough idea, we would be discussing the following Selenium WebDriver commands and their different versions:

1. **get()** methods
2. Locating links by **linkText()** and **partialLinkText()**
3. Selecting multiple items in a drop dropdown
4. Submitting a form
5. Handling iframes
6. **close()** and **quit()** methods
7. Exception Handling

#1) get() Methods

WebDriver command	**Usage**
get()	• The command launches a new browser and opens the specified URL in the browser instance • The command takes a single string type parameter that is usually a URL of application under test • To the Selenium IDE users, the command may look very much like open command driver.get("https://google.com");
getClass()	The command is used to retrieve the Class object that represents the runtime class of this object

WebDriver command	**Usage**
	driver.getClass();
getCurrentUrl()	• The command is used to retrieve the URL of the webpage the user is currently accessing • The command doesn’t require any parameter and returns a string value driver.getCurrentUrl();
getPageSource()	• The command is used to retrieve the page source of the webpage the user is currently accessing • The command doesn’t require any parameter and returns a string value • The command can be used with various string operations like contains() to ascertain the presence of the specified string value boolean result = driver.getPageSource().contains("String to find");
getTitle()	• The command is used to retrieve the title of the webpage the user is currently working on. A null string is returned if the webpage has no title • The command doesn’t require any parameter and returns a trimmed string value String title = driver.getTitle();
getText()	• The command is used to retrieve the inner text of the specified web element • The command doesn’t require any parameter and returns a string value • It is also one of the extensively used commands for verification of messages, labels, errors etc displayed on the web pages. String Text = driver.findElement(By.id("Text")).getText();

WebDriver command	Usage
getAttribute()	• The command is used to retrieve the value of the specified attribute • The command requires a single string parameter that refers to an attribute whose value we aspire to know and returns a string value as a result. driver.findElement(By.id("findID")). getAttribute("value");

getWindowHandle()	• The command is used to tackle with the situation when we have more than one window to deal with.
	• The command helps us switch to the newly opened window and performs actions on the new window.
	The user can also switch back to the previous window if he/she desires.
	
	private String winHandleBefore;
	winHandleBefore =
	driver.getWindowHandle();
	driver.switchTo().window(winHandleBefore);

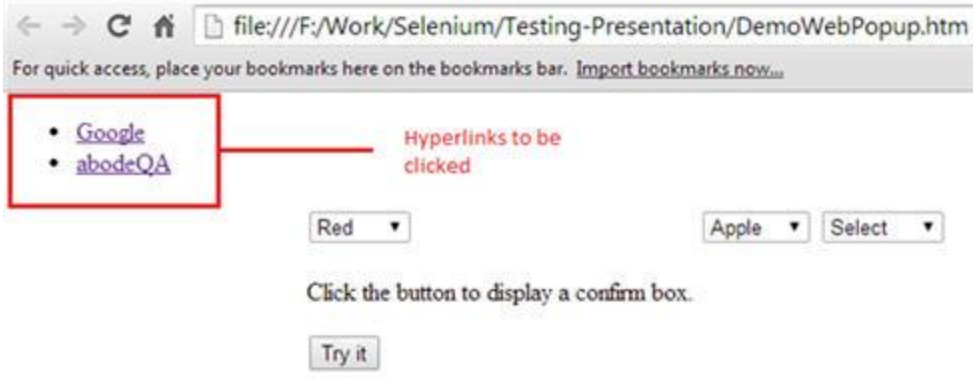
getWindowHandles()	• The command is similar to that of "getWindowHandle()" with the subtle difference that it helps to deal with multiple windows i.e. when we have to deal with more than 2 windows.

The code snippet for "getWindowHandles()" is given below:

```
public void explicitWaitForWinHandle(final WebDriver dvr, int timeOut, final boolean close) throws WeblivException
{

try {
Wait<WebDriver> wait = new WebDriverWait(dvr, timeOut);
ExpectedCondition<Boolean> condition = newExpectedCondition<Boolean>() {

@Override
public Boolean apply(WebDriver d) {
                        int winHandleNum = d.getWindowHandles().size();

if (winHandleNum > 1)
{
// Switch to new window opened
for (String winHandle : d.getWindowHandles())
{
dvr.switchTo().window(winHandle);

// Close the delete window as it is not needed
if (close && dvr.getTitle().equals("Demo Delete Window"))
{
dvr.findElement(By.name("ok")).click();
}
}
return true;
}
return false;
}
};
```

#2) Locating links by linkText() and partialLinkText()

Let us access “google.com” and “abodeqa.com” using *linkText()* and *partialLinText()* methods of WebDriver.


file:///F:/Work/Selenium/Testing-Presentation/DemoWebPopup.htm
For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...
Google
abodeQA
Hyperlinks to be clicked
Red
Apple
Select
Try it

Click the button to display a confirm box.

The above mentioned links can be accessed by using the following commands:

driver.findElement(By.linkText(“Google”)).click();

driver.findElement(By.linkText(“abodeQA”)).click();

The command finds the element using link text and then click on that element and thus the user would be re-directed to the corresponding page.

The above mentioned links can also be accessed by using the following commands:

driver.findElement(By.partialLinkText(“Goo”)).click();

driver.findElement(By.partialLinkText(“abode”)).click();

The above two commands find the elements based on the substring of the link provided in the parenthesis and thus partialLinkText() finds the web element with the specified substring and then clicks on it.

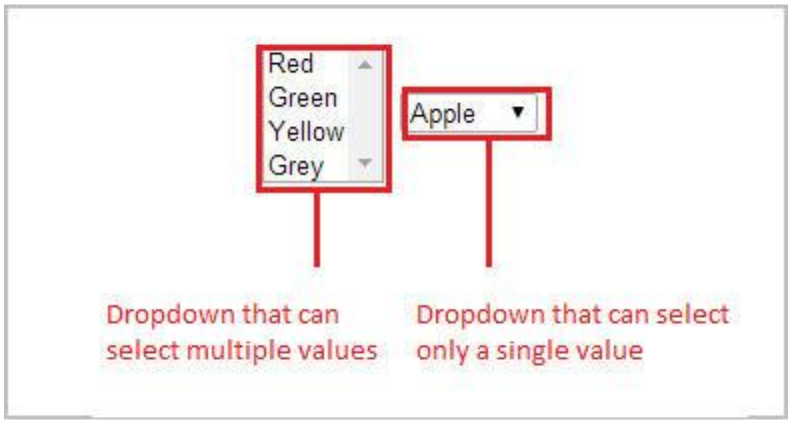
#3) Selecting multiple items in a drop dropdown

There are primarily two kinds of dropdowns:

1. **Single select dropdown**: A dropdown that allows only single value to be selected at a time.
2. **Multi select dropdown**: A dropdown that allows multiple values to be selected at a time.

Consider the HTML code below for a dropdown that can select multiple values at the same time.

```
<select id="SelectID_One" multiple="">
<option value="redvalue">Red</option
>
<option value="greenvalue">Green</option>
<option value="yellowvalue">Yellow</option>
<option value="greyvalue">Grey</option>
</select>
```

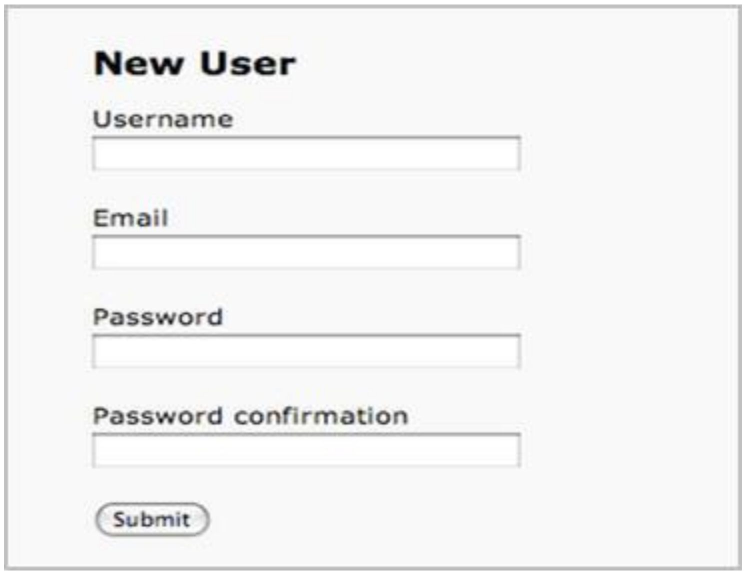

Red
Green
Yellow
Grey
Apple
Dropdown that can select multiple values
Dropdown that can select only a single value

The code snippet below illustrates the multiple selections in a drop down.

```
// select the multiple values from a dropdown
Select selectByValue = newSelect(driver.findElement(By.id("SelectID_One")));
selectByValue.selectByValue("greenvalue");
selectByValue.selectByVisibleText("Red");
selectByValue.selectByIndex(2);
```

#4) Submitting a form

Most or almost all the websites have forms that need to be filled and submitted while testing a web application. User may come across several types of forms like Login form, Registration form, File Upload form, Profile Creation form etc.



In WebDriver, user is leveraged with a method that is specifically created to submit a form. The user can also use click method to click on the submit button as a substitute to submit button.

Check out the code snippet below against the above "new user" form:

```
// enter a valid username
driver.findElement(By.<em>id</em>("username")).sendKeys("name");

// enter a valid email address
driver.findElement(By.<em>id</em>("email")).sendKeys("name@abc.com");

// enter a valid password
driver.findElement(By.<em>id</em>("password")).sendKeys("namepass");

// re-enter the password
driver.findElement(By.<em>id</em>("passwordConf")).sendKeys("namepass");

// submit the form
driver.findElement(By.<em>id</em>("submit")).submit();
```

Thus, as soon as the program control finds the submit method, it locates the element and triggers the *submit()* method on the found web element.

#5) Handling iframes

While automating web applications, there may be situations where we are required to deal with multiple frames in a window. Thus, the test script developer is required to switch back and forth between various frames or iframes for that matter of fact.

An inline frame acronym as iframe is used to insert another document with in the current HTML document or simply a web page into another web page by enabling nesting.

Consider the following HTML code having iframe within the webpage:

```
<html>
<head><title>Software Testing Help - iframe session</title>
</head
>
<body
>
<div>
<iframe id="ParentFrame">
<iframe id="ChildFrame">
<input type="text" id="Username">UserID</input>
 <input type="text" id="Password">Password</input>
</iframe>
<button id="LogIn">Log In</button>
</iframe>
</div>
</body>
</html>
```

The above HTML code illustrates the presence of an embedded iframe into another iframe. Thus, to be able to access the child iframe, user is required to navigate to the parent iframe first. After performing the required operation, user may be required to navigate back to the parent iframe to deal with the other element of the webpage.

It is impossible if a user tries to access the child iframe directly without traversing to the parent iframe first.

Select iframe by id

driver.switchTo().frame("ID of the frame");

<u>Locating iframe using tagName</u>

While locating an iframe, user might face some trouble if the iframe is not attributed with standard properties. It becomes a complex process to locate the frame and switch to it. To buckle down the situation, user is leveraged to locate an iframe using tagName method similar to the way we find any other web element in WebDriver.

driver.switchTo().frame(driver.findElements(By.tagName("iframe").get(0));

The above command locates the first web element with the specified tagName and switches over to that iframe. "get(0) is used to locate the iframe with the index value." Thus, in lines with our HTML code, the above code syntax would lead the program control to switch to "ParentFrame".

<u>Locating iframe using index:</u>

a) frame(index)

driver.switchTo().frame(0);

b) frame(Name of Frame)

driver.switchTo().frame("name of the frame");

c) frame(WebElement element)

Select Parent Window

driver.switchTo().defaultContent();

The above command brings the user back to the original window i.e. out of both the iframes.

#6) close() and quit() methods

There are two types of commands in WebDriver to close the web browser instance.

a) close(): WebDriver's close() method closes the web browser window that the user is currently working on or we can also say the window that is being currently accessed by the WebDriver. The command neither requires any parameter nor does it return any value.

b) quit(): Unlike close() method, quit() method closes down all the windows that the program has opened. Same as close() method, the command neither requires any parameter nor does it return any value.

Refer the below code snippets:

driver.close(); // closes only a single window that is being accessed by the WebDriver instance currently

driver.quit(); // closes all the windows that were opened by the WebDriver instance

#7) Exception Handling

Exceptions are the conditions or situations that halt the program execution unexpectedly.

Reasons for such conditions can be:

- Errors introduced by the user
- Errors generated by the programmer
- Errors generated by physical resources

Thus, to deal with these unexpected conditions, exception handling was conceptualized.

With respect to Java code that we implement while automating a web application can be enclosed within a block that that is capable of providing a handling mechanism against the erroneous conditions.

Catching an exception

To catch an exception, we use the below block of code

```
try{
    // Protected block
    // implement java code for automation
}
catch (ExceptionName e)
{
```

```
// catch block - Catches the exceptions generated in try block without halting the program execution
}
```

If any exception occurs in the try block/protected block, then the execution controls checks for a catch block for the matching exception type and passes the exception to it without breaking the program execution.

Multiple Catch Blocks

```
try{
      //Protected block
}
catch (ExceptionType1 e)
{
// catch block
}
catch (ExceptionType2 e)
{
// catch block
}
catch (ExceptionType3 e)
{
// catch block
}
```

In the above code, exception is likely to be caught in the first catch block if the exception type matches. If the exception type does not match, then the exception is traversed to the second catch block and third catch block and so on until the all catch blocks are visited.

WebDriver conditions and Exception Handling

When we aspire to verify the presence of any element on the webpage using various WebDriver 's conditional commands, WebDriver presumes the web element to be present on the web page. If the web element is not present on the web page, the conditional commands throw a "NoSuchElementPresentException". Thus to avoid such exceptions from halting the program execution, we use Exception Handling mechanisms. Refer the code snippet below:

```
WebElement saveButton = driver.findElement(By.id("Save"));
try{
if(saveButton.isDisplayed()){
saveButton.click();
  }
}
catch(NoSuchElementException e){
e.printStackTrace();
}
```

Conclusion

In this tutorial, we introduced various WebDriver’s commonly and excessively used commands. We tried to explain the commands with the suitable examples and code snippets.

Next Tutorial #18: In the upcoming tutorial, we would discuss about **Web tables, frames and dynamic elements** which are essential part of any web project. We will also cover the **exception handling** important topic in more details in one of the upcoming Selenium Tutorials.
Note for the Readers: Till then, stay tuned and automate the web pages and use the above documented commands.

Handling Web Tables, Frames, and Dynamic Elements in Selenium Script – Selenium Tutorial #18

In last Selenium WebDriver tutorial we learned various commonly and routinely used Selenium WebDriver commandsincluding important topics like handling iframe and exceptions in Selenium scripts.
*Moving ahead in our comprehensive series of tutorials on Selenium, in this tutorial we would discuss about **handling Web tables, iframe and dynamic elements** which are essential part of any web project.*
This tutorial consists of 3 different topics and their handling mechanisms in selenium script.

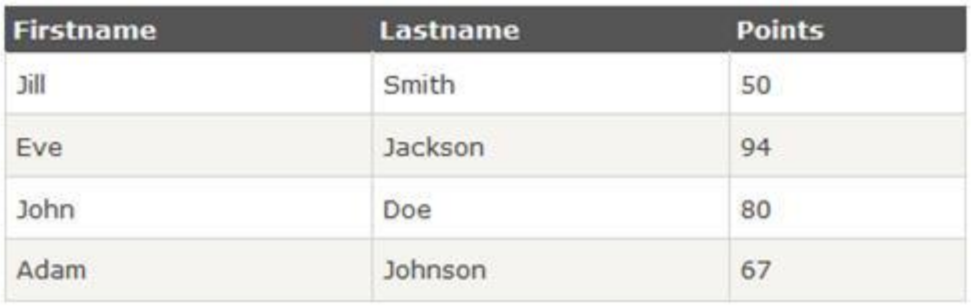
1. **Web Tables/HTML tables**
2. **Frames**
3. **Dynamic elements**

#1) Web Tables/HTML Tables

In this module we will learn about the web tables or html tables in a web page, tags available in html and how to handle web tables dynamically.

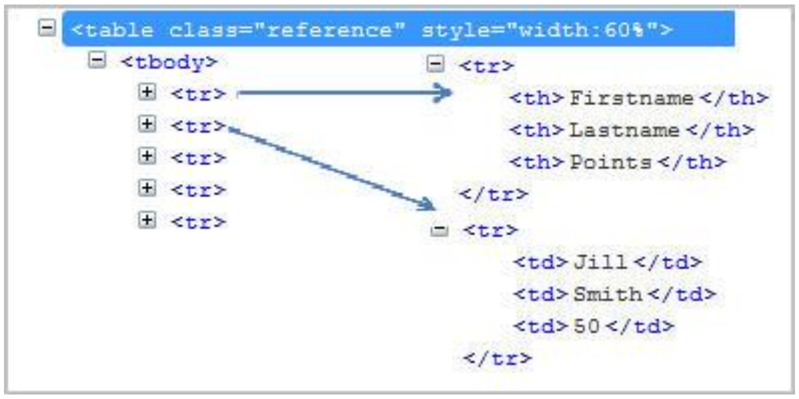
Web tables are basically group of elements that are logically stored in a row and column format. It is used to organize similar information in a web page.

Below is an example of Html table:



Firstname	Lastname	Points
Jill	Smith	50
Eve	Jackson	94
John	Doe	80
Adam	Johnson	67

Below is the snippet of html structure of an html table:



```
<table class="reference" style="width:60%">
  <tbody>
    <tr>
    <tr>
    <tr>
    <tr>
    <tr>

<tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Points</th>
</tr>
<tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
</tr>
```

Below tags are generally defined in an html tables:

1.’table’ tag defines html table.
2.’tbody’ tag defines container for rows and columns.
3.’tr’ defines rows in an html table.
4.’td’/’th’ define column of an html table.

Find the details of a web table:

There are many ways we can handle a web table.

Approach #1:

Below is the xpath of one of the cell in html table. Let’s say “firstname”

//div[@id=’main’]/table[1]/tbody/tr[1]/th[1]

tr[1] defines first row and th[1] defines first column.

If number of rows and columns are always constant, let’s say our html table will always have 5 rows and 3 columns.

```
for(int numberOfRows=1; numberOfRows<=5; numberOfRows++)
{
for(int numberOfCol=1; numberOfCol <=3; numberOfCol++)
{
System.out.println(driver.findElement(By.xpath
(“//div[@id='main']/table[1]/tbody/tr
[“+numberOfRows+”]/th[“+numberOfCol+”]”)));
}
}
```

Except row and column number, each component of xpath remains the same. So you can iterate using “for loop” for each row and column as mentioned above.

Approach #2:

First approach is best suitable for the table which doesn’t change its dimensions and always remains the same. Above approach will not be a perfect solution for dynamically changing web tables.

Let’s take above html table as an example:

```
WebElement htmltable=driver.findElement(By.xpath("//*[@id='main']/table[1]/tbody"));
List<WebElement> rows=htmltable.findElements(By.tagName("tr"));

for(int rnum=0;rnum<rows.size();rnum++)
{
List<WebElement> columns=rows.get(rnum).findElements(By.tagName("th"));
System.out.println("Number of columns:"+columns.size());

 for(int cnum=0;cnum<columns.size();cnum++)
{
System.out.println(columns.get(cnum).getText());
}
}
```

Step 1: First get the entire html table and store this in a variable ‘htmltable’ of type web element.
Step 2: Get all the rows with tag name ‘tr’ and store all the elements in a list of web elements. Now all the elements with tag ‘tr’ are stored in ‘rows’ list.
Step 3: Loop through each row and get the list of elements with tag *‘th’*. *‘rows.get(0)’* will give first row and *‘findElements(By.tagName(“th”))’* will give list of columns for the row.
Step 4: Iterate using *‘columns.getsize()’* and get the details of each cell.
Note: Above approach will be best suitable if the table dimensions changes dynamically.
This concludes the topic how to handle web tables in selenium. Next we will learn about handling an element inside frame.

#2) Frames:

In this section we will learn about the frames in a web page and how to identify the frames. Also we will find out how we can handle a frame in selenium WebDriver.

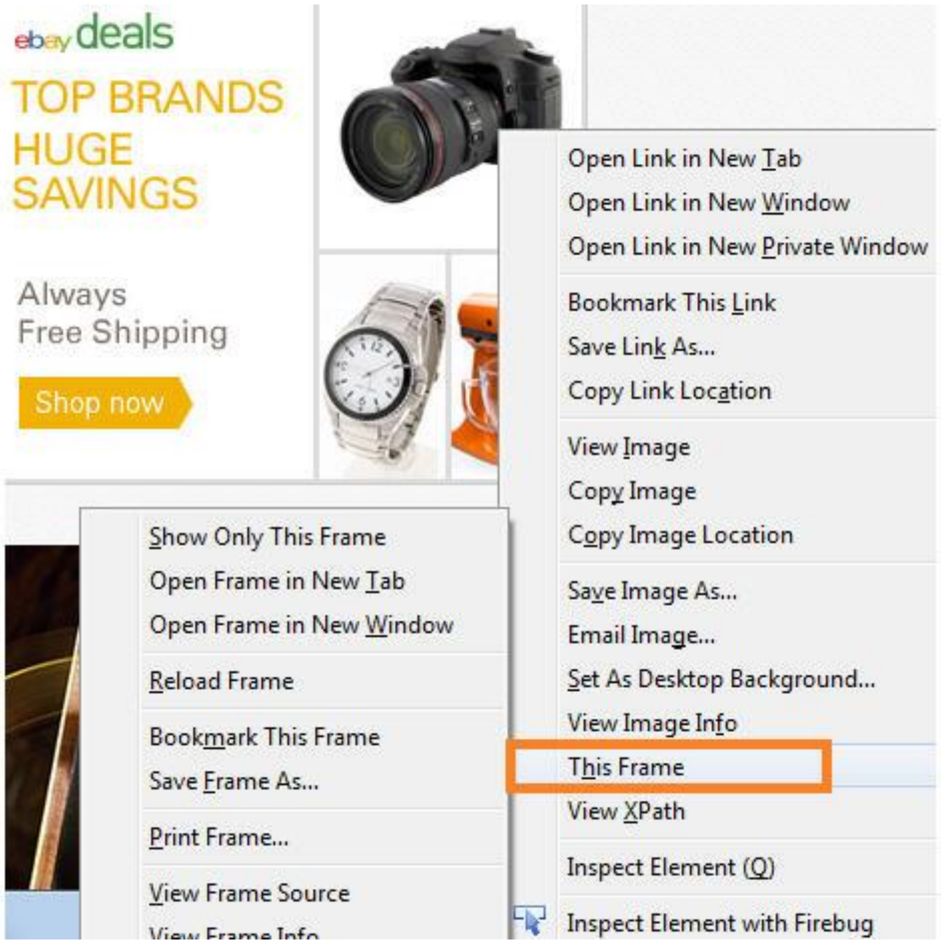
Many developers like to place elements inside frame. Frame is just like a container where few elements can be grouped.

Identification of a frame:

Different ways to know if the element is present inside a frame or not

#1. Right click on the element. Check if “This Frame” option is available. If This frame option is available, it means that the element is inside a frame.

#2. View page source of the web page and check if any tag is available for ‘iframe’.



Verify Number of frames in a webpage:
All the frames are having tag name as “iframe”.

List<WebElement> frameList=driver.findElements(By.tagName(“iframe”));
System.out.println(frameList.size());
In above example: *frameList* will have all the list of frames and *frameList.size()* will give the number of frames.
Handling an element inside frame:
If an element is inside a frame then control has to switch to frame first and then start operating on the elements.

Step 1: To switch inside a frame:
driver.switchTo().frame(1); //pass frame number as parameter.
or
driver.switchTo().frame(“frame Name”); //pass frame name as parameter.
or
driver.switchTo().frame(“xpath of the frame”);
Step 2: After switching inside a frame selenium will be able to operate on elements.

driver.findElement(//[@id='username']).sendKeys("username");*
driver.findElement(//[@id='pass']).sendKeys("password");*
Here, we have learned how to handle an element inside frame and next we will cover about the different ways to handle dynamic element.

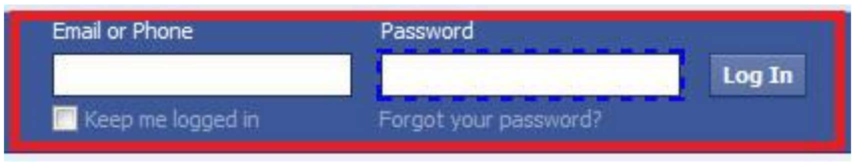
#3) Dynamic elements:
In this section we will learn different ways to handle dynamic element and construct generic Xpath.

In few scenarios, element attributes change dynamically. It can be 'id', 'name' etc.

Example: let's say 'id' of a username field is 'username_123' and the xpath will be *//*[@id='username_123']* but when you open the page again the 'id' of 'username' field might have changed and the new value may be 'username_234'.
In this case the test will fail because the selenium could not find the xpath you have passed earlier as the id of the field has changed to some other value.

There are many approaches depending upon the type of problem:

Problem Type 1: If part of the attribute value changes**.**
Example: As in the above example, id value changes but few fields remains constant.
'username_123' changed to 'username_234' but 'username' always remained constant.
You can construct xpath as below:
driver.findElement(By.xpath("//[contains(@id,'username')]")).sendKeys("username");*
driver.findElement(By.xpath("//[starts-with(@id,'user')]")).sendKeys("username");*
'contains' is a java method which checks if id contains the substring username.
starts-with() checks if any attribute starts with "user".
Problem Type 2: If entire value of the attribute changes dynamically.
Again in this case, there could be different approaches:



For example: if id of 'login' field changes dynamically and there is no constant value to use contains method.
Solution: Use of sendKeys.
Selenium provides different api to use function keys. For example tab key, enter keys, F5 etc.
***Step 1**: Enter password*
driver.findElement(By.id("password")).sendKeys("password"));

Step 2*: Use key functions to navigate to element.*
driver.findElement(By.id("password")).sendKeys(Keys.ENTER));
or
driver.findElement(By.id("password")).sendKeys(Keys.TAB));

Conclusion:

Web tables, frames and dynamic elements are essential part of any web project. It is always desirable to write effective code to handle web tables and dynamic elements.

Understanding the construction of generic xpath which is very helpful while handling dynamic elements. In case of a frame, your script has to switch the frame and then operate on the element.

Next tutorial #19: In next Selenium tutorial we will learn **about types of exceptions and how to handle exceptions in java in Selenium scripts.**
Please post your queries related to Web tables, frames and handling dynamic element if you have any.

Handling Exceptions Using Exception Handling Framework in Selenium Scripts – Selenium Tutorial #19

In last WebDriver tutorial we learned about 3 different types of important web elements like Web Tables, Frames and Dynamic elementsand their handling mechanisms in selenium script
Before moving ahead with Framework tutorials in this Selenium training series, here in this tutorial we will learn about **types of exceptions and how to handle exceptions in java and Selenium scripts**.
Developers/testers use exception handling framework to handle exception in selenium scripts.

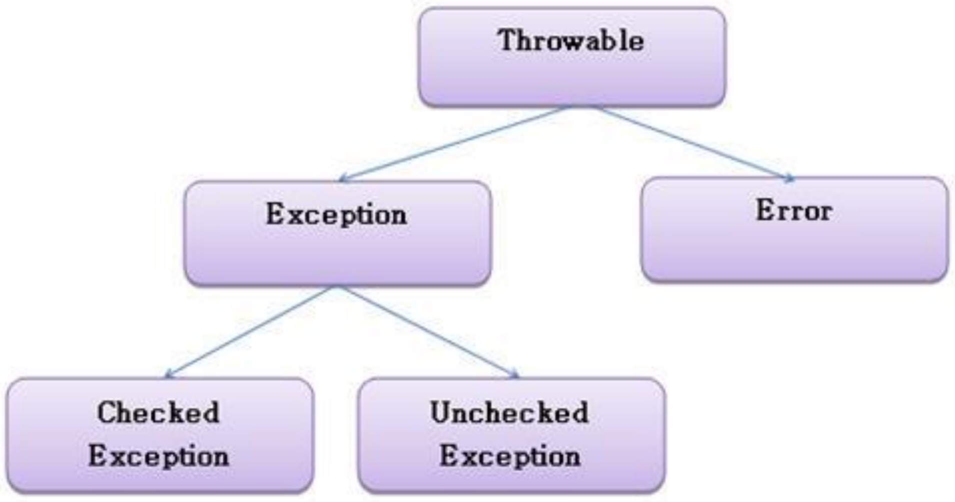
Example: When selenium script fails due to wrong locator, then developer should be able to understand the reason for failure and this can be achieved easily if the exception is handled properly in the program.
Below we have described the types of exceptions and the different ways how we can use exception handling framework in selenium scripts.

Exceptions are events due to which java program ends abruptly without giving expected output. Java provides a framework where user can handle exceptions.

There are three kinds of exceptions:

1. Checked Exception
2. Unchecked Exception
3. Error

Class hierarchy of exception and error:



Class hierarchy of exception and error

#1) Checked Exception: Checked exception is handled during compile time and it gives compilation error if it is not caught and handled during compile time.

Example: *FileNotFoundException*, *IOException* etc.

#2) Unchecked Exception: In case of unchecked exception, compiler does not mandate to handle. Compiler ignores during compile time.

Example: *ArrayIndexoutOfBoundException*

#3) Error: When a scenario is fatal and program cannot recover then JVM throws an error. Errors cannot be handled by try catch block. Even if user tries to handle error by using Try catch block, it cannot recover from error.

Example: *Assertion error*, *OutOfMemoryError* etc.

Exception handling:

Try and Catch block:

try-catch blocks are generally used to handle exceptions. Type of exceptions is declared in catch block which is expected to come. When an exception comes in try block, immediately control moves to catch block.

Example:

```
try {
   br = new BufferedReader(new FileReader("Data"));
   } catch(IOException ie)
   {
       ie.printStackTrace();
   }
```

There can be multiple catch blocks for one try block depending upon type of exception.

Example:

```
try {
   br = new BufferedReader(new FileReader("Data"));
   } catch(IOException ie)
   {
     ie.printStackTrace();
   } catch(FileNotFoundException file){
     file.printStackTrace();
   }
```

***throws* Exception:**

throws keyword in java is used to throw an exception rather than handling it. All checked exceptions can be thrown by methods.

Example:

```
public static void main(String[] args) throws IOException
{
BufferedReader br=new BufferedReader(newFileReader("Data"));
```

```
    while ((line = br.readLine()) != null)
        {
          System.out.println(line);
        }
}
```

***finally* block:**

finally block executes irrespective of execution of try catch block and it executes immediately after try/catch block completes.

Basically file close, database connection etc. can be closed in finally block.

Example:

```
try {
   br = new BufferedReader(new FileReader("Data"));
   } catch(IOException ie)
   {
     ie.printStackTrace();
   }
Finally {
        br.close();
      }
```

In the above example, *BufferReader* stream is closed in finally block. *br.close()* will always execute irrespective of execution of try and catch block.

Note: finally block can exist without any catch block. It is not necessary to have a catch block always.

There can be many catch blocks but only one finally block can be used.

Throwable: Throwable is parent class for error and exception. Generally it is difficult to handle errors in java. If programmer is not sure about the type of error and exception, then it is advised to use Throwable class which can catch both error and exception.

Example:

```
try {
  br = new BufferedReader(new FileReader("Data"));
    } catch (Throwable t)
    {
      t.printStackTrace();
    }
```

Exceptions in Selenium WebDriver:

Selenium has its own set of exceptions. While developing selenium scripts, programmer has to handle or throw those exceptions. Below are few examples of exceptions in selenium.

Examples:

ElementNotVisibleException: If selenium tries to find an element but element is not visible within page

NoAlertPresentException: If user tries to handle an alert box but alert is not present.
NoSuchAttributeException: While trying to get attribute value but attribute is not available in DOM.
NoSuchElementException: This exception is due to accessing an element which is not available in the page.
WebDriverException: Exception comes when code is unable to initialize WebDriver.

Conclusion:

Exception handling is the essential part of every java program as well as selenium script. We can build robust and optimal code by **handling exception in smart ways**. And it is also a best practice to handle exceptions in script which will give you a better report when program fails due to any reason.
Here we have tried to cover the process and framework of exception handling which is required to be implemented in selenium scripts.

Remember it is not mandatory to always handle exception in *try-catch* block. You can also throw an exception depending upon the requirement in script.
Next Tutorial #20: In the upcoming tutorial, we would discuss about the **various types of testing frameworks available**. We would also study about the pros and cons of using a fledged framework approach in automation testing. We would discuss in detail about Test data driven framework.
Please post your queries, related to handling exception in Selenium WebDriver, if you have any

Selenium Framework

- Tutorial #20 – Most popular Test Automation frameworks(Must Read)
- Tutorial #21 – Selenium Framework Creation & Accessing Test Data from Excel (Must Read)
- Tutorial #22 – Creating Generics and Testsuite
- Tutorial #23 – Using Apache ANT
- Tutorial #24 – Setting up Selenium Maven Project
- Tutorial #25 – Using Hudson Continuous integration tool

Most Popular Test Automation Frameworks with Pros and Cons of Each – Selenium Tutorial #20

In the last few Selenium tutorials, we discussed about various commonly and popularly used commands in WebDriver, handling web elements like Web Tables, Framesand handling exceptions in Selenium scripts. We discussed each of these commands with sample code snippets and examples so as to make you capable of using these commands effectively whenever you are encountered with similar situations. Amongst the commands we discussed in the previous tutorial, few of them owe utmost importance.

As we move ahead in the Selenium series, we would concentrate our focus towards **Automation Framework creation** in the next few upcoming tutorials. We would also shed light on various aspects of an Automation framework, types of Automation frameworks, benefits of using a framework and the basic components that constitutes an Automation framework.

What is Framework?
A framework is considered to be a combination of set protocols, rules, standards and guidelines that can be incorporated or followed as a whole so as to leverage the benefits of the scaffolding provided by the Framework.

Let us consider a real life scenario.
We very often use lifts or elevators. There are a few guidelines those are mentioned within the elevator to be followed and taken care off so as to leverage the maximum benefit and prolonged service from the system.

Thus, the users might have noticed the following guidelines:

- Keep a check on the maximum capacity of the elevator and do not get onto an elevator if the maximum capacity has reached.
- Press the alarm button in case of any emergency or trouble.
- Allow the passenger to get off the elevator if any before entering the elevator and stand clear off the doors.
- In case of fire in the building or if there is any haphazard situation, avoid the use of elevator.
- Do not play or jump inside the elevator.
- Do not smoke inside the elevator.
- Call for the help/assistance if door doesn't open or if the elevator doesn't work at all. Do not try to open the doors forcefully.

There can be many more rules or sets of guidelines. Thus, these guidelines if followed, makes the system more beneficial, accessible, scalable and less troubled for the users.

Now, as we are talking about "Test Automation Frameworks", let us move our focus towards them.

Test Automation Framework

A "Test Automation Framework" is scaffolding that is laid to provide an execution environment for the automation test scripts. The framework provides the user with various benefits that helps them to develop, execute and report the automation test scripts efficiently. It is more like a system that has created specifically to automate our tests.

In a very simple language, we can say that a framework is a constructive blend of various guidelines, coding standards, concepts, processes, practices, project hierarchies, modularity, reporting mechanism, test data injections etc. to pillar automation testing. Thus, user can follow these guidelines while automating application to take advantages of various productive results.

The advantages can be in different forms like ease of scripting, scalability, modularity, understandability, process definition, re-usability, cost, maintenance etc. Thus, to be able to grab these benefits, developers are advised to use one or more of the Test Automation Framework.

Moreover, the need of a single and standard Test Automation Framework arises when you have a bunch of developers working on the different modules of the same application and when we want to avoid situations where each of the developer implements his/her approach towards automation.

Note: Take a note that a testing framework is always application independent that is it can be used with any application irrespective of the complications (like Technology stack, architecture etc.) of application under test. **The framework should be scalable and maintainable.**

Advantage of Test Automation framework

1. Reusability of code
2. Maximum coverage
3. Recovery scenario
4. Low cost maintenance
5. Minimal manual intervention
6. Easy Reporting

Types of Test Automation Framework

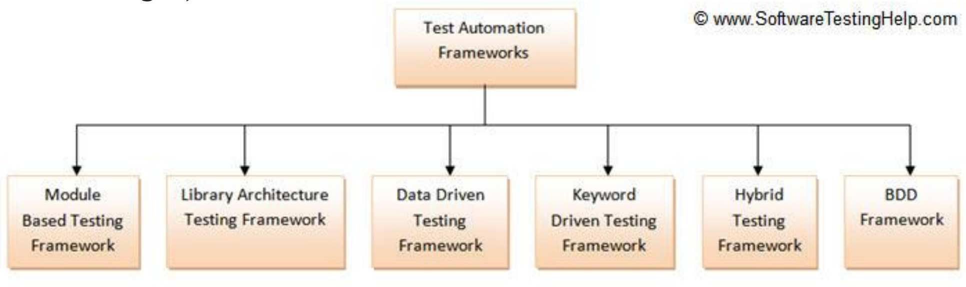
Now that we have a basic idea of what is an Automation Framework, in this section we would harbinger you with the various types of Test Automation Frameworks those are available in the market place. We would also try shed lights over their pros and cons and usability recommendations.

There is a divergent range of Automation Frameworks available now days. These frameworks may differ from each other based on their support to different key factors to do automation like reusability, ease of maintenance etc.

Let us discuss the few most popularly used Test Automation Frameworks:

1. Module Based Testing Framework
2. Library Architecture Testing Framework
3. Data Driven Testing Framework
4. Keyword Driven Testing Framework
5. Hybrid Testing Framework
6. Behavior Driven Development Framework

(click on image to view enlarged)


© www.SoftwareTestingHelp.com
Test Automation Frameworks
Module Based Testing Framework
Library Architecture Testing Framework
Data Driven Testing Framework
Keyword Driven Testing Framework
Hybrid Testing Framework
BDD Framework

Test Automation Frameworks

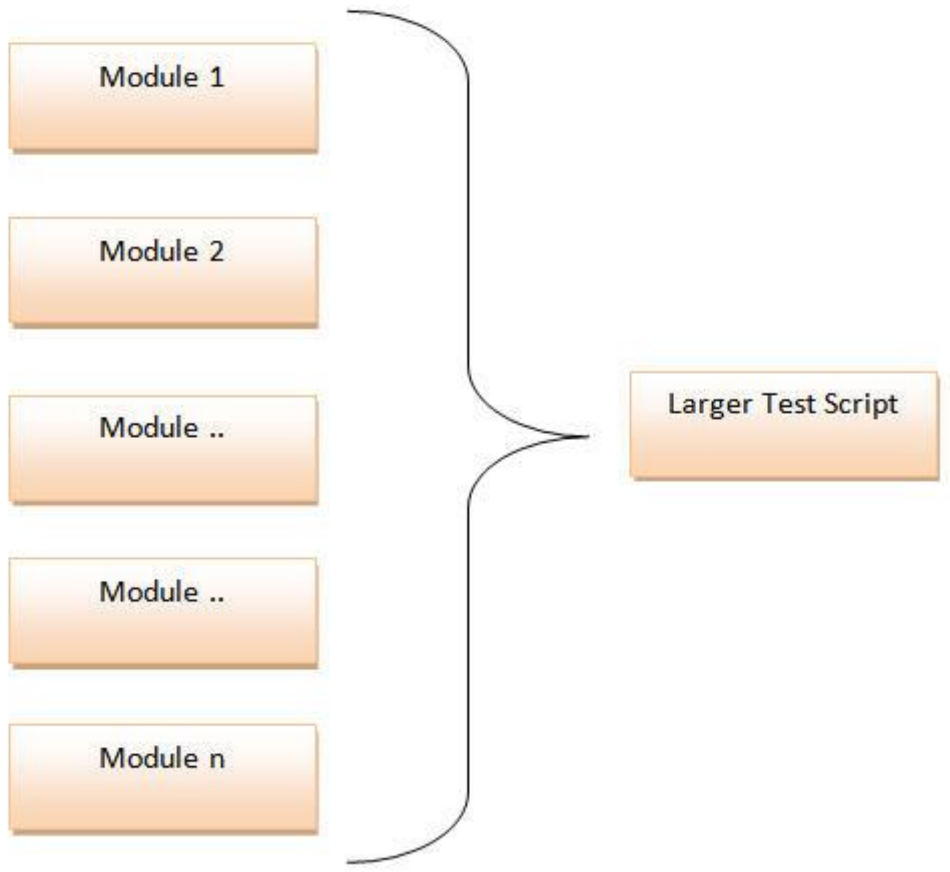
Let us discuss each of them in detail.

But before that I would also like to mention that despite having these framework, user is always leveraged to build and design his own framework which is best suitable to his/her project needs.

#1) Module Based Testing Framework

Module based Testing Framework is based on one of the popularly known OOPs concept – Abstraction. The framework divides the entire "Application Under Test" into number of logical and isolated modules. For each module, we create a separate and independent test script. Thus, when these test scripts taken together builds a larger test script representing more than one modules.

These modules are separated by an abstraction layer in such a way that the changes made in the sections of the application doesn't yields affects on this module.



Pros:

1. The framework introduces high level of modularization which leads to easier and cost efficient maintenance.
2. The framework is pretty much scalable
3. If the changes are implemented in one part of the application, only the test script representing that part of the application needs to be fixed leaving all the other parts untouched.

Cons:

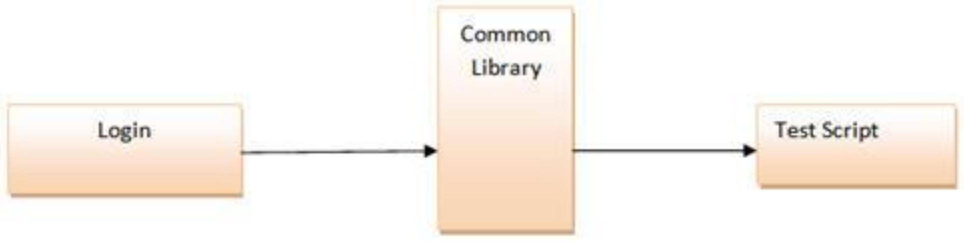
1. While implementing test scripts for each module separately, we embed the test data (Data with which we are supposed to perform testing) into the test scripts. Thus, whenever we are supposed to test with a different set of test data, it requires the manipulations to be made in the test scripts.

#2) Library Architecture Testing Framework

The Library Architecture Testing Framework is fundamentally and foundationally built on Module Based Testing Framework with some additional advantages. Instead of dividing the application under test into test scripts, we segregate the application into functions or rather common functions can be used by the other parts of the application as well. Thus we create a common library constituting of common functions for the application under test. Therefore, these libraries can be called within the test scripts whenever required.

The basic fundamental behind the framework is to determine the common steps and group them into functions under a library and call those functions in the test scripts whenever required.

Example: The login steps can be combined into a function and kept into a library. Thus all the test scripts those require to login the application can call that function instead of writing the code all over again.



Pros:

1. Like Module Based Framework, this framework also introduces high level of modularization which leads to easier and cost efficient maintenance and scalability too.
2. As we create common functions that can be efficiently used by the various test scripts across the Framework. Thus, the framework introduces a great degree of re-usability.

Cons:

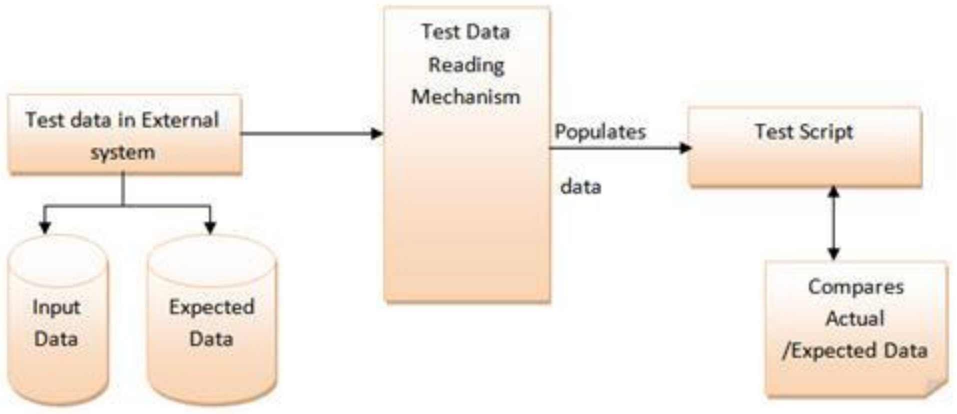
1. Like Module Based Framework, the test data is lodged into the test scripts, thus any change in the test data would require changes in the test script as well.
2. With the introduction of libraries, the framework becomes a little complicated.

#3) Data Driven Testing Framework

While automating or testing any application, at times it may be required to test the same functionality multiple times with the different set of input data. Thus, in such cases, we can’t let the test data embedded in the test script. Hence it is advised to retain test data into some external data base outside the test scripts.

Data Driven Testing Framework helps the user segregate the test script logic and the test data from each other. It lets the user store the test data into an external database. The external databases can be property files, xml files, excel files, text files, CSV files, ODBC repositories etc. The data is conventionally stored in “Key-Value” pairs. Thus, the key can be used to access and populate the data within the test scripts.

Note: The test data stored in an external file can belong to the matrix of expected value as well as matrix of input values.

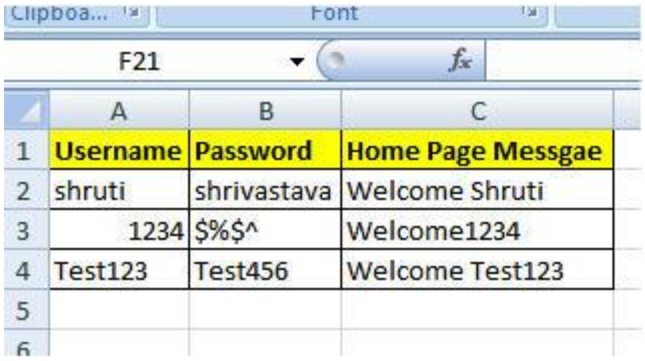

Test data in External system
Input Data
Expected Data
Test Data Reading Mechanism
Populates data
Test Script
Compares Actual /Expected Data

<u>Example</u>:

Let us understand the above mechanism with the help of an example.

Let us consider the “Gmail – Login” Functionality.

Step 1: First and the foremost step are to create an external file that stores the test data (Input data and Expected Data). Let us consider an excel sheet for instance.



	A	B	C
1	Username	Password	Home Page Messgae
2	shruti	shrivastava	Welcome Shruti
3	1234	$%$^	Welcome1234
4	Test123	Test456	Welcome Test123
5			
6			

Step 2: The next step is to populate the test data into Automation test Script. For this purpose several API’s can be used to read the test data.

```
public void readTD(String TestData, String testcase) throws Exception {
        TestData=readConfigData(configFileName,"TestData",driver);
        testcase=readConfigData(configFileName,"testcase",driver);
              FileInputStream td_filepath = new FileInputStream(TestData);
              Workbook td_work =Workbook.getWorkbook(td_filepath);
              Sheet td_sheet = td_work.getSheet(0);
              if(counter==0)
              {
       for (int i = 1,j = 1; i <= td_sheet.getRows()-1; i++){
                              if(td_sheet.getCell(0,i).getContents().equalsIgnoreCase(testcase)){
          startrow = i;
                 arrayList.add(td_sheet.getCell(j,i).getContents());
                 testdata_value.add(td_sheet.getCell(j+1,i).getContents());}}
        for (int j = 0, k = startrow +1; k <= td_sheet.getRows()-1; k++){
               if(td_sheet.getCell(j,k).getContents()==""){
                     arrayList.add(td_sheet.getCell(j+1,k).getContents());
                     testdata_value.add(td_sheet.getCell(j+2,k).getContents());}}
               }
               counter++;
}
```

The above method helps to read the test data and the below test step helps the user to type in the test data on the GUI.

element.sendKeys(obj_value.get(obj_index));

Pros:

1. The most important feature of this framework is that it considerably reduces the total number of scripts required to cover all the possible combinations of test scenarios. Thus lesser amount of code is required to test a complete set of scenarios.
2. Any change in the test data matrix would not hamper the test script code.
3. Increases flexibility and maintainability
4. A single test scenario can be executed altering the test data values.

Cons:

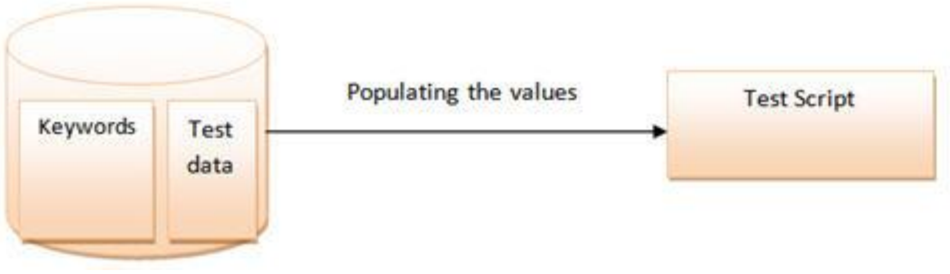
1. The process is complex and requires an extra effort to come up with the test data sources and reading mechanisms.
2. Requires proficiency in a programming language that is being used to develop test scripts.

#4) Keyword Driven Testing Framework

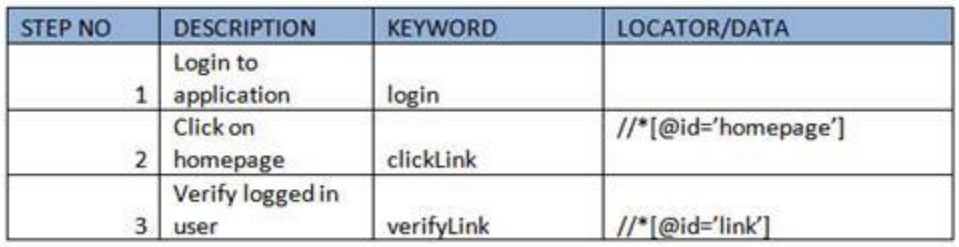
The Keyword driven testing framework is an extension to Data driven Testing Framework in a sense that it not only segregates the test data from the scripts, it also keeps the certain set of code belonging to the test script into an external data file.

These set of code are known as Keywords and hence the framework is so named. Key words are self-guiding as to what actions needs to be performed on the application.

The keywords and the test data are stored in a tabular like structure and thus it is also popularly regarded as Table driven Framework. Take a notice that keywords and test data are entities independent of the automation tool being used.



<u>Example</u> Test case of Keyword Driven Test Framework



STEP NO	DESCRIPTION	KEYWORD	LOCATOR/DATA
1	Login to application	login	
2	Click on homepage	clickLink	//*[@id='homepage']
3	Verify logged in user	verifyLink	//*[@id='link']

In the above example keywords like login, clickLink and verifyLink are defined within the code. Depending upon the nature of application keywords can be derived. And all the keywords can be reused

multiple times in a single test case. Locator column contains the locator value that is used to identify the web elements on the screen or the test data that needs to be supplied.
All the required keywords are designed and placed in base code of the framework.

Pros:

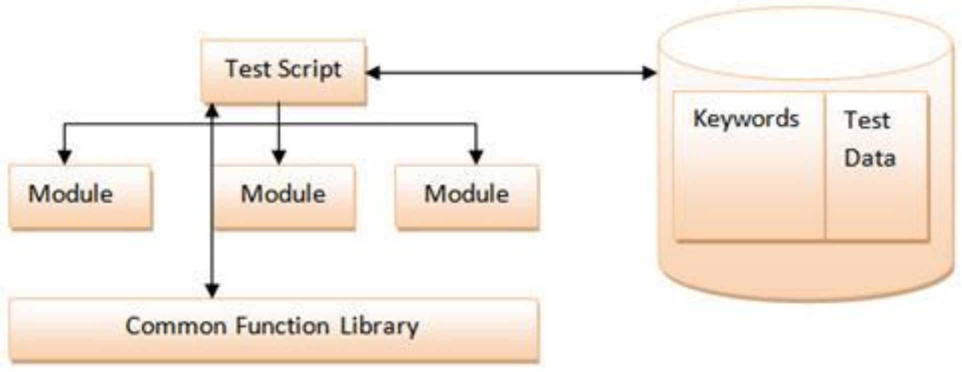
1. In addition to advantages provided by Data Driven testing, Keyword driven framework doesn't require the user to possess scripting knowledge unlike Data Driven Testing.
2. A single keyword can be used across multiple test scripts.

Cons:

1. The user should be well versed with the Keyword creation mechanism to be able to efficiently leverage the benefits provided by the framework.
2. The framework becomes complicated gradually as it grows and a number of new keywords are introduced.

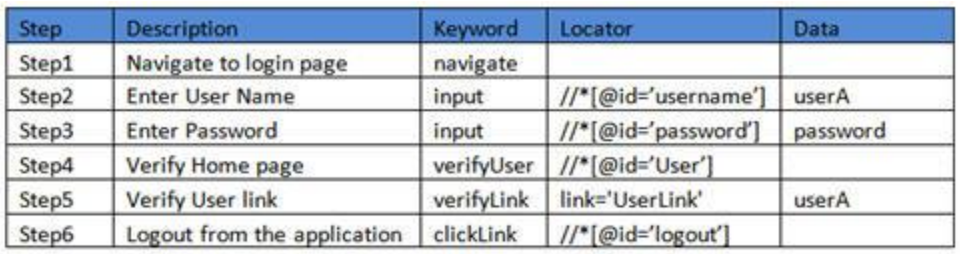
#5) Hybrid Testing Framework

As the name suggests, the Hybrid Testing Framework is a combination of more than one above mentioned frameworks. The best thing about such a setup is that it leverages the benefits of all kinds of associated frameworks.



<u>Example</u> of Hybrid Framework

Test sheet would contain both the keywords and the Data.



Step	Description	Keyword	Locator	Data
Step1	Navigate to login page	navigate		
Step2	Enter User Name	input	//*[@id='username']	userA
Step3	Enter Password	input	//*[@id='password']	password
Step4	Verify Home page	verifyUser	//*[@id='User']	
Step5	Verify User link	verifyLink	link='UserLink'	userA
Step6	Logout from the application	clickLink	//*[@id='logout']	

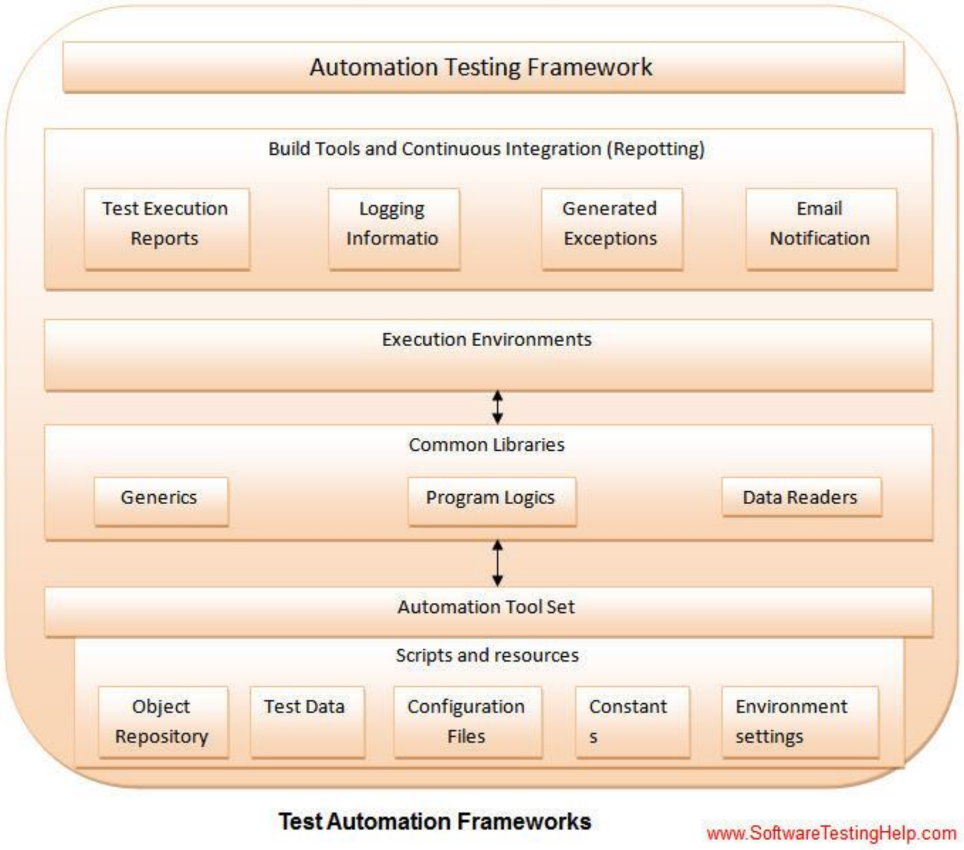
In the above example, keyword column contains all the required keywords used in the particular test case and data column drives all the data required in the test scenario. If any step does not need any input then it can be left empty.

#6) Behavior Driven Development Framework

Behavior Driven Development framework allows automation of functional validations in easily readable and understandable format to Business Analysts, Developers, Testers, etc. Such frameworks do not necessarily require the user to be acquainted with programming language. There are different tools available for BDD like cucumber, Jbehave etc. Details of BDD framework are discussed later in Cucumber tutorial. We have also discussed details on Gherkin language to write test cases in
Cucumber.

Components of Automation Testing Framework

(click on image to view enlarged)



Though the above pictorial representation of a framework is self-explanatory but we would still highlight a few points.

1. **Object Repository**: Object Repository acronym as OR is constituted of the set of locators types associated with web elements.
2. **Test Data:** The input data with which the scenario would be tested and it can be the expected values with which the actual results would be compared.
3. **Configuration File/Constants/ Environment Settings**: The file stores the information regarding the application URL, browser specific information etc. It is generally the information that remains static throughout the framework.

4. **Generics/ Program logics/ Readers**: These are the classes that store the functions which can be commonly used across the entire framework.
5. **Build tools and Continuous Integration**: These are the tools that aids to the frameworks capabilities to generate test reports, email notifications and logging information.

Conclusion

The frameworks illustrated above are the most popular frameworks used by the testing fraternity. There are various other frameworks also in the place. For all the further tutorials we would base on the **Data Driven Testing Framework**.

In this tutorial, we discussed about the basics of an Automation Framework. We also discussed about the types of frameworks available in the market.

Next Tutorial #21: In the next tutorial, we would briefly **introduce you with the sample framework, the MS Excel which would store the test data, excel manipulations etc.**

Till then feel free to ask your queries about automation frameworks.

Selenium Framework Creation and Accessing Test Data from Excel – Selenium Tutorial #21

In the last tutorial, we familiarized you with the **basics of test automation Frameworks**, its components and types. The frameworks illustrated in the previous tutorial were a few amongst the most popular frameworks used by the testing fraternity.
We briefly discussed about Module based Frameworks, Library Architecture based framework, Keyword driven framework, Data driven Framework and Hybrid Framework. There are various other frameworks also in the place.

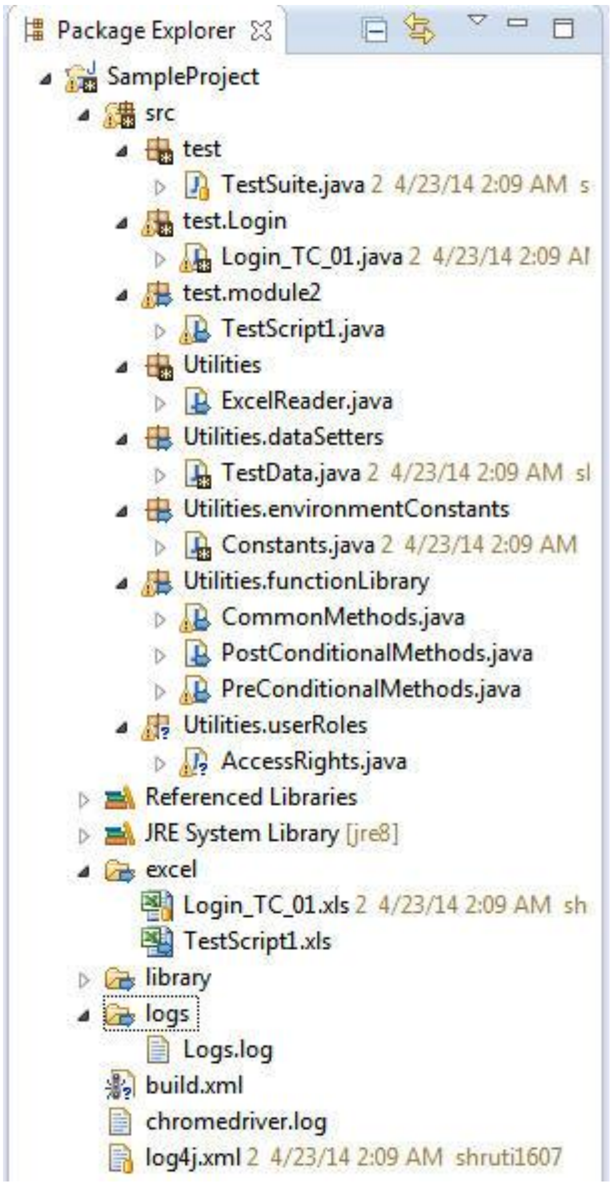
*Please take a note that we would be adopting **Data Driven Test Automation Framework for the rest of our tutorials.***

In the current tutorial in this series, we would make you acquainted with a **sample framework, the Excels which would store the test data and their Excel manipulations**. On the same lines, we would move forward and introduce new strategies and resources to mature our framework.
So let's learn:

- Framework creation strategy using a sample project
- Access the test data stored in the external data source

Moving ahead, we would start with the description of the project hierarchy that we would be creating in order to segregate the various project components.

Refer the below image for the project hierarchy created for the sample project. The below java project can be easily created within the eclipse the way we have created the projects in the earlier tutorials.



```
Package Explorer
SampleProject
  src
    test
      TestSuite.java 2  4/23/14 2:09 AM  s
    test.Login
      Login_TC_01.java 2  4/23/14 2:09 AI
    test.module2
      TestScript1.java
    Utilities
      ExcelReader.java
    Utilities.dataSetters
      TestData.java 2  4/23/14 2:09 AM  sl
    Utilities.environmentConstants
      Constants.java 2  4/23/14 2:09 AM
    Utilities.functionLibrary
      CommonMethods.java
      PostConditionalMethods.java
      PreConditionalMethods.java
    Utilities.userRoles
      AccessRights.java
  Referenced Libraries
  JRE System Library [jre8]
  excel
    Login_TC_01.xls 2  4/23/14 2:09 AM  sh
    TestScript1.xls
  library
  logs
    Logs.log
  build.xml
  chromedriver.log
  log4j.xml 2  4/23/14 2:09 AM  shruti1607
```

Selenium Project Folder Structure – Walkthrough

#1) src – The folder contains all the test scripts, generics, readers and utilities. All these resources are nothing but the simple java classes. Under the source (src) folder, we have created a hierarchy of folders.

a) test – The "test" folder is constituted of majorly two ingredients – testsuite and the folders representing the various modules of the application under test. Thus, each of these folders contains the test scripts specific to the module to which it is associated. Testsuite is a logical combination of more than one test scripts. Thus, the user can mark an entry of any of the test script within the testsuite that he/she desires to execute in the subsequent runs.

b) utilities – The "utilities" folder is constituted of various generics, constants, Readers and classes for implementing user defined exceptions. Each of the folders under utilities has got its own significance.

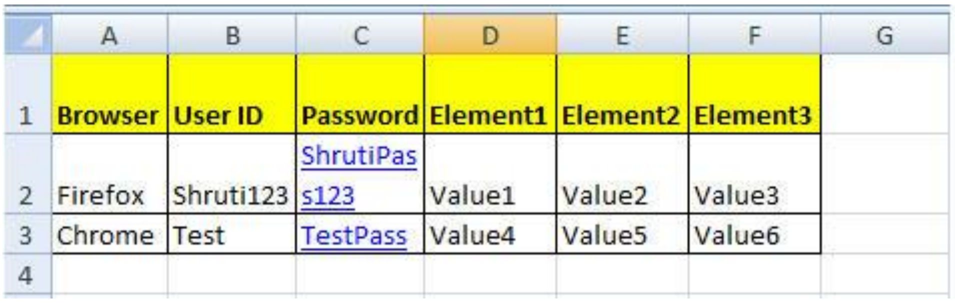
- **Excel Reader –** A generic and common class has been created to read the test data (input parameters and expected results) from the Excel sheets

- **EnvironmentConstants –** The folder are integration of the java classes that stores the static variables referencing to the paths and other environmental details. These details can be Application URL, URL to the Databases, Credentials for Databases, and URL to any third party tool being used. The disparate application URLs can be set for different environments (dev, prod, test, master, slave etc).
- **DataSetters –** The folder incorporates the classes that implement the getters and setters of the test data fetched from the Excels. To lode multiple sets of Test data, we create ArrayLists.
- **UserRoles –** The folder accommodates the classes that take care of the Role based access criteria if any for instinct users.
- **FunctionLibrary –** The folder is constituted of the classes which contain functions and methods that can be shared and used amongst the multiple classes. Very often, we are suppose to perform certain procedures prior and aftermath to the actual test execution like login to the application, setting up environments, activities related to rolls, data manipulations, writing results, methods those generate pre/post-conditions to other methods. Since we tend to perform these activities for all or most of the test script. Thus it is always recommended to create a separate class for such activities instead of coding them repeatedly in each of the test script.
  - **PreConditionalMethods**
  - **PostConditionalMethods**

Very often, we are suppose to perform certain procedures prior and aftermath to the actual test execution like login to the application, setting up environments, activities related to user rolls, data manipulations, writing results, methods those generate pre/post-conditions to other methods. Since we tend to perform these activities for all or most of the test script, thus it is always recommended to create a separate class for such activities instead of coding them repeatedly in each of the test script.

CommonMethods
Like Pre and post conditions, there may be methods and functions those can be used by more than one test script. Thus, these methods are grouped together in a class. The testscript can access these methods using the object of the common class.

#2) excelFiles – The excel files are considered to be the data source/data providers for test script execution. These files store the test data into key value pairs. Make a note that we create a separate excel sheet for each of the test script i.e. each test script has its own test data file. The name of the test script and the corresponding test data files/ excel sheet has been kept same for the traceability perspective. Check out the sample test data format below:
Test Data Format

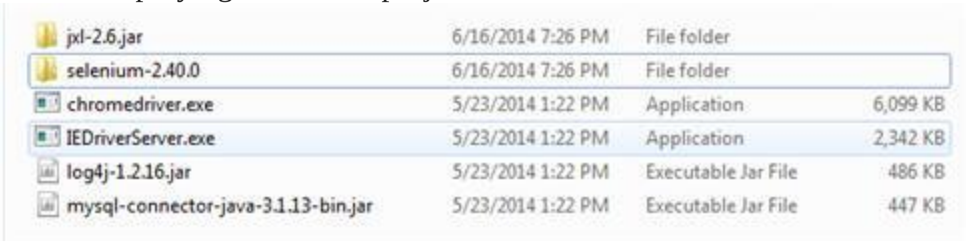


	A	B	C	D	E	F	G
1	Browser	User ID	Password	Element1	Element2	Element3	
2	Firefox	Shruti123	ShrutiPass123	Value1	Value2	Value3	
3	Chrome	Test	TestPass	Value4	Value5	Value6	
4							

Each of the columns represents a key and each of the rows represents a test data/value. Specify the multiple rows in order to execute the same test script with multiple data sets.

Mark that the test data formats are solely user defined. Thus based on your requirements, you can customize the test data files.

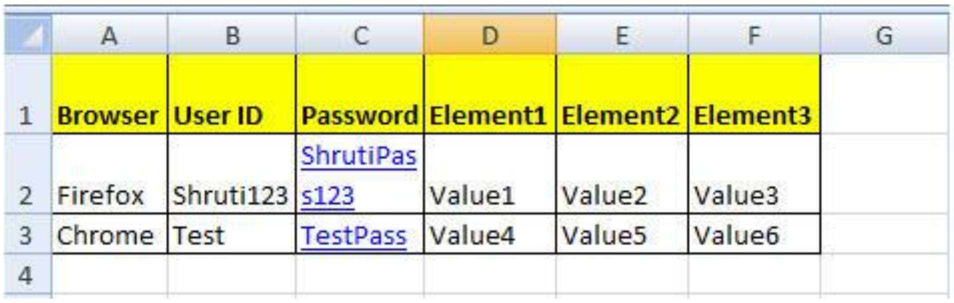
#3) library – The folder acts as a repository/artifactory for all the required jar files, libraries, drivers etc to successfully build the test environment and to execute the test scripts. Refer the following figure to check out the libraries we would be employing within our project.



jxl-2.6.jar	6/16/2014 7:26 PM	File folder	
selenium-2.40.0	6/16/2014 7:26 PM	File folder	
chromedriver.exe	5/23/2014 1:22 PM	Application	6,099 KB
IEDriverServer.exe	5/23/2014 1:22 PM	Application	2,342 KB
log4j-1.2.16.jar	5/23/2014 1:22 PM	Executable Jar File	486 KB
mysql-connector-java-3.1.13-bin.jar	5/23/2014 1:22 PM	Executable Jar File	447 KB

#4) logs – The folder contains a .txt file that stores the logging information upon each execution.
#5) testMaterial – The folder contains the actual test data that needs to be uploaded if any. This folder would come into picture when we come across test scenarios where the user is required to upload files, documents, pictures, reports etc.
#6) build.xml – The xml file is used by the “Ant Server” to automate the entire build process.
#7) log4j.xml – This xml file is used by a Java based utility named as “Log4j” to generate the execution logs.
Note: We would study more about the logs, user defined exceptions and Ant in detail in the upcoming tutorials. So don’t panic if you get confused between the notions.
Now, as we move forward let us understand the phenomenon where we access the excel files and populate the test data into our test scripts.
In order to comprehend the process easily, we would break down the process into the following steps.

Test Data Creation
Step 1: The first and the foremost step is to create the test data with which we would be executing the test scripts. Considering the aforementioned test data format, let us create an excel file named as “TestScript1”. Furnish the values in the elements.



	A	B	C	D	E	F	G
1	Browser	User ID	Password	Element1	Element2	Element3	
2	Firefox	Shruti123	ShrutiPass123	Value1	Value2	Value3	
3	Chrome	Test	TestPass	Value4	Value5	Value6	
4							

Step 2: The next step is to download a standard java based API/Library named as "Java excel Library" (jxl) to be able to access the already created generic methods for Excel Manipulation.

Step 3: Create a generic excel reader class named as "ExcelReader.java". Copy the below code in the ExcelReader.java.

```
package Utilities;
import java.io.File;
import java.io.IOException;
import java.util.Hashtable;
import jxl.Sheet;
import jxl.Workbook;
import jxl.read.biff.BiffException;

/**
 * This is a utility class created to read the excel test data file before performing the test steps.
 * This class loads the excel file and
 * reads its column entries.
 *
 */

public class ExcelReader {
            /**
             * The worksheet to read in Excel file
             */

            public static Sheet wrksheet;
            /**
             * The Excel file to read
             */

            public static Workbook wrkbook = null;
            /**
```

```
	 * Store the column data
	 */

	public static Hashtable<String, Integer> dict = new Hashtable<String, Integer>();
	/**
	 * Create a Constructor
	 *
	 * @param ExcelSheetPath
	 * @throws BiffException
	 * @throws WeblivException
	 */

	public ExcelReader(String ExcelSheetPath) throws IOException, BiffException {

		// Initialize
		  try {
				wrkbook = Workbook.getWorkbook(new File(ExcelSheetPath));
				wrksheet = wrkbook.getSheet("Sheet1");
		  } catch (IOException e) {
				 throw newIOException();
		  }
	}
	/**
	 * Returns the Number of Rows
	 *
	 * @return Rows
	 */

	public static int RowCount() {
		  return wrksheet.getRows();
	}
	/**
	 * Returns the Cell value by taking row and Column values as argument
	 *
	 * @param column
	 * @param row
	 * @return Cell contents
	 */

	public static String ReadCell(int column, int row) {
		returnwrksheet.getCell(column, row).getContents();
	}
	/**
	* Create Column Dictionary to hold all the Column Names
	 */
	public static void ColumnDictionary() {
		// Iterate through all the columns in the Excel sheet and store the
		// value in Hashtable
		for (int col = 0; col < wrksheet.getColumns(); col++) {
			dict.put(ReadCell(col, 0), col);
		}
	}
	/**
	 * Read Column Names
	 *
	 * @param colName
	 * @return value
	 */

	public static int GetCell(String colName) {
		try {
			int value;
			value = ((Integer) dict.get(colName)).intValue();
			returnvalue;
		} catch(NullPointerException e) {
			return(0);
		}
	}
}
```

Step 4: Create a generic class –"CommonMethods.java". Create a common method within the class that would read the cells from the excel sheet using the methods implemented in ExcelReader.java.

```
/**
* Read the test data from excel file
*
* @param data The TestData data object
*/

public void readExcelData (TestData data) {
    ArrayList<String> browser = new ArrayList<String>();
    ArrayList<String> username = new ArrayList<String>();
                                        ArrayList<String> password = new ArrayList<String>();
     ArrayList<String> element1 = new ArrayList<String>();
     ArrayList<String> element2 = new ArrayList<String>();
     ArrayList<String> element3 = new ArrayList<String>();

     // Get the data from excel file
     for (int rowCnt = 1; rowCnt < ExcelReader.RowCount(); rowCnt++) {
     browser.add(ExcelReader.ReadCell(ExcelReader.GetCell("Browser"), rowCnt));
     username.add(ExcelReader.ReadCell(ExcelReader.GetCell("User ID"), rowCnt));
                 password.add(ExcelReader.ReadCell(ExcelReader.GetCell("Password"), rowCnt));
                 element1.add(ExcelReader.ReadCell(ExcelReader.GetCell("Element1"), rowCnt));
                 element2.add(ExcelReader.ReadCell(ExcelReader.GetCell("Element2"), rowCnt));
      element3.add(ExcelReader.ReadCell(ExcelReader.GetCell("Element3"), rowCnt));
     }
     data.setBrowser(browser);
     data.setLoginUser(username);
     data.setPassword(password);
     data.setElement1(element1);
     data.setElement2(element2);
     data.setElement3(element3);
     }
```

Step 5: Create a new java class named as "TestData.java". This class would act as a getter and setter for excel data. Copy and paste the following code in the TestData.java class.

```
package Utilities.dataSetters;
import java.util.ArrayList;
public class TestData {
    private ArrayList<String> loginUser = null;
    private ArrayList<String> password = null;
    private ArrayList<String> browser = null;
    private ArrayList<String> element1 = null;
    private ArrayList<String> element2 = null;
     private ArrayList<String> element3 = null;
     /**
      * @return loginUser
      */
     public ArrayList<String> getLoginUser() {
           return loginUser;
     }
     /**
      * @param loginUser
      */
     public void setLoginUser(ArrayList<String> loginUser) {
           this.loginUser = loginUser;
     }
     /**
      * @return password
      */
     public ArrayList<String> getPassword() {
           return password;
     }
     /**
      * @param password
      */
     public void setPassword(ArrayList<String> password) {
           this.password = password;
     }
     /**
      * @return browser
      */
     public ArrayList<String> getBrowser() {
           return browser;
     }
     /**
      * @param browser
      */
     public void setBrowser(ArrayList<String> browser) {
           this.browser = browser;
     }
     /**
      * @return element1
      */
     public ArrayList<String> getElement1() {
           return element1;
     }
     /**
      * @param element1
```

```
	*/
	public void setElement1(ArrayList<String> element1) {
		this.element1 = element1;
	}
	/**
	 * @return element2
	 */
	public ArrayList<String> getElement2() {
		return element2;
	}
	/**
	 * @param element2
	 */
	public void setElement2(ArrayList<String> element2) {
		this.element2 = element2;
	}
	/**
	 * @return element3
	 */
	public ArrayList<String> getElement3() {
		return element3;
	}
	/**
	 * @param element3
	*/
	public void setElement3(ArrayList<String> element3) {
		this.element3 = element3;
	}
}
```

<u>Step 6</u>: The next step is to create instances of “TestData.java” and “CommonMethods.java” java classes within the test script in order to access and populate the test data. Refer the below code snippet for object initialization, reading excel data and populating the values wherever required.

```
// Create Objects
public ExcelReader excelReaderObj;
CommonMethods commonMethodobj = new CommonMethods();
TestData td = new TestData();

// Load the excel file for testing
excelReaderObj = new ExcelReader(Path of the excel);

```

```
// Load the Excel Sheet Col in to Dictionary for use in test cases
excelReaderObj.ColumnDictionary();

// Get the data from excel file
commonMethodobj.readExcelData (td);

 // Populate the username
driver.findElement(By.id("idofElement")).sendKeys(data.getLoginUser().get(0));
```

Therefore using the instance of testData.java class in conjunction with getters, any test data value can be populated within the script.

Conclusion:

The tutorial mainly revolved around the notions like Framework Creation and Accessing test data from the excels. We made you acquainted with the Framework creation strategy using a sample project. We briefly laid light on the various components and aspects of our framework.

In order to access the test data stored in the external data source, we used a java based API – jxl. We also created the sample code for reading and populating the excel data into the testscripts.

Next Tutorial #22: In the next tutorial, we would base our tutorial on the **concepts of generics and their accessibility mechanism**. We would create a few sample generic methods and then access them within the test scripts. We would also introduce you with the concept of Testsuite and the sample code development.

Creating Generics and Testsuites – Selenium Tutorial #22

In the previous tutorial, we started off with the representation of the sample project hierarchy and various framework components. We also discussed about the data source – "excels" used to store the test data and their excel manipulations. We also discussed about the new strategies and resources to mature our framework.
Now we are moving ahead with advanced topics in this Selenium Training Series. In this session, we would take the opportunity to discuss two important concepts that plays an important role to mature the framework. We would discuss the **concept of Generics and reusability aspects**. We would also discuss about **creation and significance of Test suite**.
For better understanding we would accompany the concepts with adequate examples and sample code.

Generics

By the literal notion, a generic is something that can act as a descriptive of an entire group or classes.
While automating applications, we come across various end to end scenarios. An end to end scenario may consist of several trivial functionality. Thus, many of these functionality can act as common functionality to more than one test script with slight or almost no modifications.

Hence, it is advisable to create a generic class consisting of methods that can be claimed as common and can be shared among multiple test scripts instead of implementing the same code again and again for multiple test scripts.

Take a note that generics also introduce the power of reusability in our framework. Reusability reduces time taken for code, errors, bugs, maintenance etc. exceptionally.

Type of Generics

#1) Application Specific

The meager functionality belonging to application under test can become a part of the Application Specific generics. Take the Login Functionality for instance. Login is one such functionality that can be a fragment of almost all the test scripts. Thus, instead of writing the login code all over again in the test scripts, we would create a common method in the generic class and call it wherever needed.

#2) Framework Specific

Aside from Application specific generics, we may have common methods which do not directly relate with the application under test but are part of the chosen framework. Consider an Excel reading functionality when we have employed Test Data Driven Framework. It would make no sense if we would write the code for reading excels again and again in all the test scripts. Therefore, we induce the code once in the generic class and make a call to it whenever required.

Creation of Generic Class

User is leveraged to create as many generic classes as he/she desires based on the modularity infused.

Let us understand the concept of generic by creating one.

Step 1: Create a new java class “CommonMethods.java” that would act as a generic class consisting of common methods preferably in the package other than where test scripts reside.
Step 2: The next step is to copy and paste the below code in the “CommonMethods.java” generic class. Number of common methods can be implemented inside the periphery of this class. Below is the code snippet for login functionality.

```
/**
* Login the Test application
*
* @param username
* @param password
*/
public void login(String username, String password) {
try {
// Enter User Name
            WebElement userName = driver.findElement(By.id("loginID"));
userName.clear();
userName.sendKeys(username);
// Enter Password
WebElement passWord = driver.findElement(By.id("Password"));
passWord.clear();
passWord.sendKeys(password);
// Click on the Sign In Button
WebElement signin = driver.findElement(By.id("SignIn_button"));
signin.click();
driver.manage().window().maximize();
} catch (Exception e) {
e.printStackTrace();
}
}
```

Take a note that the aforementioned method is a parameterized method. Thus, the same method can be used to test the login functionality with different sets of test data.

Step 3: The next step is to call the common method within the test script. The process is a two step process. First we create the instance of the generic class within the test class and then we call the common method on the created instance by passing the required arguments. In the code fragment below, we created an instance of "TestScript1.java" class and called the login () method to login the application.

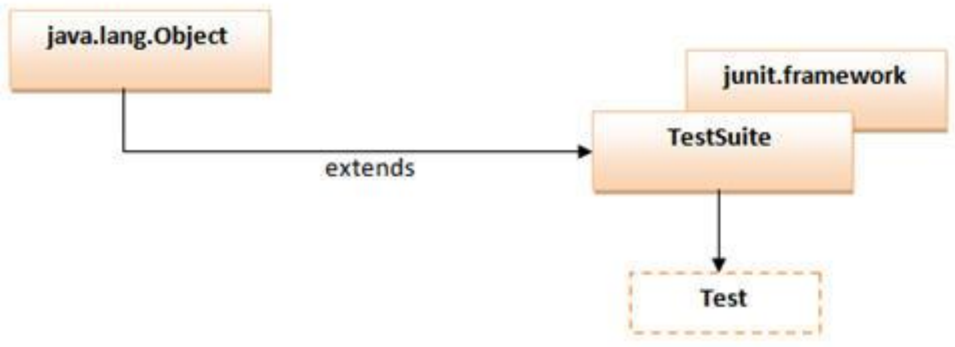
```
// Create Object of the generic class
toolsObj = new Tools();
// Login the test application by calling the common method
preTestObj.login("username", "password");
```

Take a mention that the above code can be placed anywhere inside the test class. User can place the code in the setup () method or in the test () method.

Testsuite

Test suite is an assortment of more than one test script grouped together for execution purpose. Thus, test suite executes the number of specified test scripts unattended. Test suite has the capability to cite the test scripts to be executed automatically; all that is required from the user is to mark an entry each for the individual test script within the test suite. The entry is supposed to be the "class name" of the test script with ".class" extension or simply the compiled form of our java class.

Below is the sample Test suite created in java. Take a note that Test suite is a java based class that belongs to the family of JUnit. Thus, you may encounter several JUnit annotations in the code.



Code Snippet

```
package com.axway.webliv.tests;
import org.junit.AfterClass;
import org.junit.BeforeClass;
import org.junit.runner.JUnitCore;
import org.junit.runner.Result;
import org.junit.runner.RunWith;
import org.junit.runner.notification.Failure;
import org.junit.runners.Suite;
 import com.axway.webliv.tests.MetaData.*;
```

```

@RunWith(Suite.class)
@Suite.SuiteClasses({

    ChangeStatusBinarySphereTest.class,
    ChangeStatusRestrictionTest.class,
    ChangeStatusDocSphereTest.class
,
   })

public class TestSuite {
    /**
     * Setup method to set system properties
     */
    @BeforeClass
    public static void Setup() {
    }
    /**
     * @param args
     */
    public static void main(String[] args) {
        Result result = JUnitCore.runClasses(TestSuite.class);
        System.out.println("TEST CASES RUN: " + result.getRunCount());
        System.out.println("TEST CASES FAILED: " + result.getFailureCount());
        for (Failure failure : result.getFailures()) {
            System.out.println("\nTEST NAME: " + failure.getTestHeader());
            System.out.println("\nERROR: " + failure.getMessage() + "\n");
            System.out.println(failure.getTrace());
            System.exit(1);
        }
    }
    /**
     * Report test results
     */
    @AfterClass
    public static void TearDown() {
    }
}
```

Code Walk-Through

A test suite is none other than a simple JUnit class having *setup()* and *teardown()* methods; the one's we discussed at length in our preceding tutorials. The only remarkable difference lies in its competence to execute more than one test script in a single go.

Import Statements

```
import org.junit.AfterClass;
import org.junit.BeforeClass;
import org.junit.runner.JUnitCore;
import org.junit.runner.Result;
import org.junit.runner.RunWith;
import org.junit.runner.notification.Failure;
import org.junit.runners.Suite;
```

The above import statements are embedded in the class to be able to use various annotations provided by the JUnit.

import org.junit.runner.JUnitCore;
import org.junit.runner.RunWith;
import org.junit.runners.Suite;
The above statements possess the underlying architecture to execute the test suite consisting of multiple test classes.

import org.junit.runner.Result;
The import statement allows the user to store the test execution statuses and their manipulations.

import org.junit.AfterClass;
import org.junit.BeforeClass;
These import statements are used to identify and annotate *setup()* and *teardown()* methods. The *setup()* method annotated with BeforeClass instructs the program control to execute the method before each of the test script execution. Like *setup()*, *teardown()* method annotated with AfterClass tells the program control to execute the method after each of the test script execution.

Class Entry

```
@RunWith(Suite.<strong>class</strong>)
@Suite.SuiteClasses({
    ChangeStatusBinarySphereTest.<strong>class</strong>,
    ChangeStatusRestrictionTest.<strong>class</strong>,
    ChangeStatusDocSphereTest.<strong>class</strong>,
    })
```

This section of the class allows the user to mark the entries of the test script to be executed in the next run. Remember the entries are marked with “.class” extension i.e. in their compiled formats.

Execution – main ()

```
public static void main(String[] args) {
Result result = JUnitCore.runClasses(TestSuite.class);
System.out.println("TEST CASES RUN: " + result.getRunCount());
System.out.println("TEST CASES FAILED: " + result.getFailureCount());
for (Failure failure : result.getFailures()) {
System.out.println("\nTEST NAME: " + failure.getTestHeader());
System.out.println("\nERROR: " + failure.getMessage() + "\n");
System.out.println(failure.getTrace());
 System.exit(1);
}
}
```

This part of the code deals with the execution. The program execution always initiates from the main().

The runClasses method of JUnitCore class is used to execute the test suite. The “Result class” and its methods are used to determine the execution status in terms of Passed and Failed test cases.

Thus, user is leveraged to play around with the test suite class to be able to suffice his/her requirements.

Conclusion

In this tutorial, we tried to make you acquainted with the concept of generics and common methods. We also discussed the benefits we get out generics like reusability. We also shared the practical approaches towards creation of generics and their accessibility.

Here are the cruxes of this article:

- Generic is something that can act as a descriptive of an entire group or classes. Generic in our framework is a class that is solely consists of methods those can be shared across multiple test classes.
- Generics can be classified into two categories:
  - Application Specific
  - Framework Specific
- A simple java class can be created to act as a Generic. Number of common methods can be implemented inside the generic class. These methods can be parameterized methods.
- The common methods can be accessed by calling them on the instance of generic class within the test scripts.
- Test suite is an assortment of more than one test script grouped together for execution purpose. Thus, test suite executes the number of specified test scripts unattended.
- A test suite is none other than a simple JUnit class having *setup()* and *teardown()* methods; the one’s we discussed at length in our preceding tutorials. The only remarkable difference lies in its competence to execute more than one test script in a single go.

Next Tutorial #23: Going forward in the next tutorial we would study about **yet another tool for automating the entire build process. Thus, we would discuss “Ant” in the next tutorial at length**. We would discuss about the need of an hour to use a build tool in Test Automation. We would slide down to the deeper sections where we would define the project dependencies and create build.xml file.

Note for the Readers – As we have already covered the major part of framework in this and the previous few tutorials, readers can start exercising these concepts and can come up with their own customized framework.

Apache ANT – a Tool for Automating Software Build Processes and its Importance in Testing – Selenium Tutorial #23

In the last tutorial, we tried to make you acquainted with the concept of generics and common methods. We also discussed the benefits we get out of generics like reusability. We also shared the practical approaches towards creation of generics and their accessibility.
In the current tutorial in this **Selenium automation** series, we would shed light on **a build tool named as "Apache Ant"**. We would broadly discuss its applicability and importance besides the practical approach.
Take a note that the tutorial is limited to testing aspects of using Apache Ant.
Apache Ant is a very popular and conventional build tool of our times. Ant is an open source java based build tool provided by Apache Software Foundation freely distributed under GNU license. Apache Ant plays a significant role in developer's as well as Tester's day to day work schedule. The tool has immense power to build the development code into deployment utilities.
Ant is a tool that automates the software building process. Ant is not just limited to compilation of code, rather packaging, testing and a lot more can be achieved in some simple steps.
The tool works on the principle of targets and dependencies defined in the XML files. Ant libraries are used to build the applications. The libraries have a set of defined tasks to archive, compile, execute, document, deploy, and test and many more targets. Moreover, Ant allows the user to create his/her own tasks by implementing their own libraries.

Ant is primarily used with Java Applications but it can still be used for applications built on other languages depending on the extended support.

The most important aspect of using Ant is that it doesn't demands another set of code to be written in order to build the application, rather the entire process is defined by targets which are none other than XML elements.

Apache Ant Benefits

- **Ease of Use** – The tool provides a wide range of tasks that almost fulfills all the build requirements of the user.
- **Platform Independent** – Ant is written in Java thus is a platform independent build tool. The only requirement for the tool is JDK.
- **Extensibility** – As the tool is written in Java and the source code is freely available, user is leveraged with the benefit to extend the tool's capabilities by writing java code for adding task in Ant Libs.

Apache Ant Features

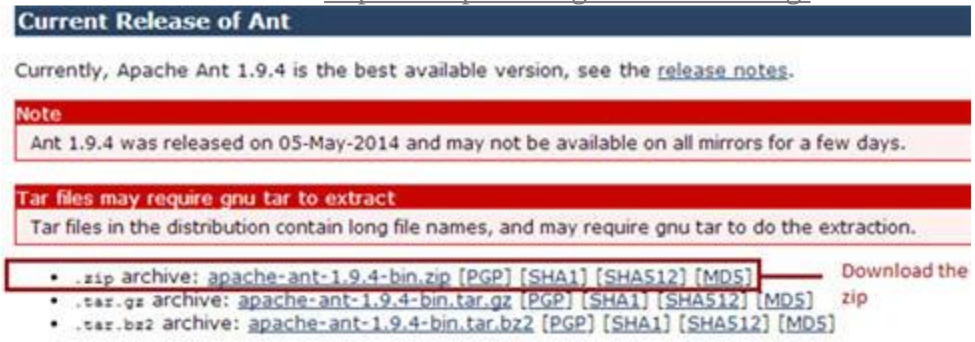
- Can compile java based applications
- Can create Java Doc
- Can create war, jar, zip, tar files
- Can copy files to at different locations
- Can delete or move files
- Can send Emails to the stakeholders
- Supports Junit 3, Junit 4, TestNG etc.
- Can convert XML based test reports to HTML reports
- Can make directories
- Can check out the code from version control system (SVN, GIT, CVS etc).
- Can execute test scripts and test suites

Environment Setup

Let us demonstrate the entire setup process step by step.

Step 1: Apache Ant Download

The first and the foremost step is to download the zipped folder of Apache Ant latest version from the repository. The distribution is available at "http://ant.apache.org/bindownload.cgi".



Current Release of Ant

Currently, Apache Ant 1.9.4 is the best available version, see the release notes.

Note

Ant 1.9.4 was released on 05-May-2014 and may not be available on all mirrors for a few days.

Tar files may require gnu tar to extract

Tar files in the distribution contain long file names, and may require gnu tar to do the extraction.

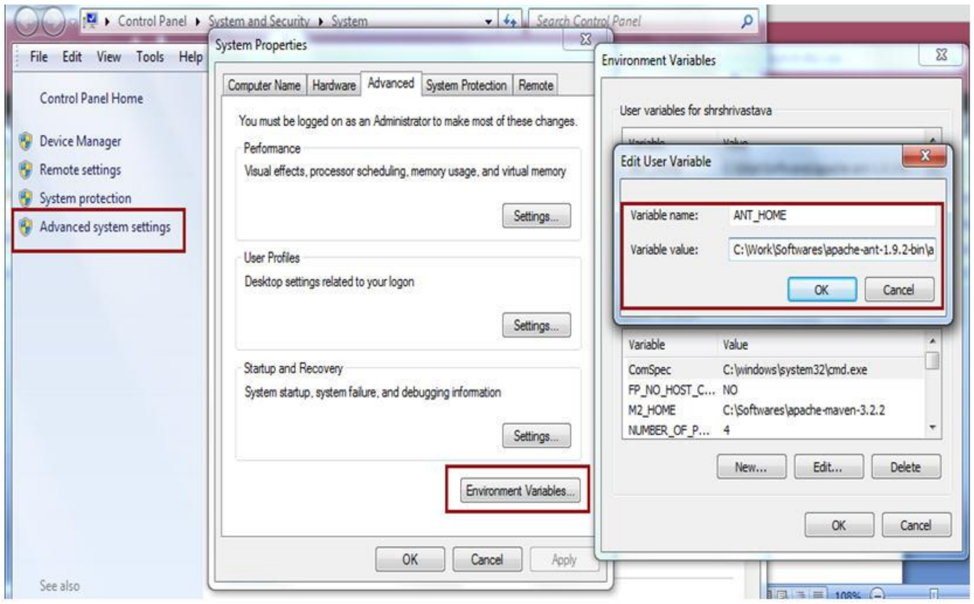
- .zip archive: apache-ant-1.9.4-bin.zip [PGP] [SHA1] [SHA512] [MD5]
- .tar.gz archive: apache-ant-1.9.4-bin.tar.gz [PGP] [SHA1] [SHA512] [MD5]
- .tar.bz2 archive: apache-ant-1.9.4-bin.tar.bz2 [PGP] [SHA1] [SHA512] [MD5]

Step 2: Extract folder and Set Environment Variables

Extract the zipped folder at any desired location onto local file system.

Prior to setting up environment for Ant, it is required to install and set JDK on to your system. I am assuming that the JDK is already set and installed, Thus moving forward with the Ant Setup.

Create an environment variable for "ANT_HOME" and set the variable's value to the location of Ant folder. Refer the following screenshot for the same.



You must be logged on as an Administrator to make most of these changes.

Visual effects, processor scheduling, memory usage, and virtual memory

Desktop settings related to your logon

System startup, system failure, and debugging information

User variables for shrshrivastava

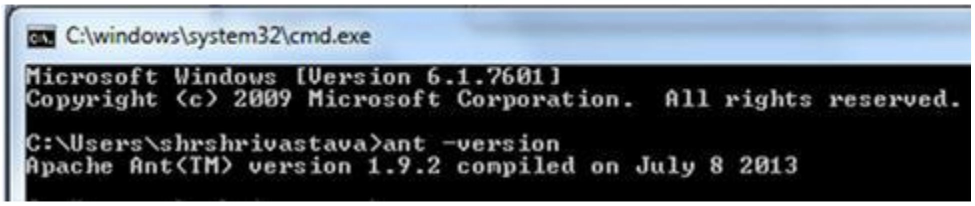
Variable name: ANT_HOME

Variable value: C:\Work\Softwares\apache-ant-1.9.2-bin\a

Variable	Value
ComSpec	C:\windows\system32\cmd.exe
FP_NO_HOST_C...	NO
M2_HOME	C:\Softwares\apache-maven-3.2.2
NUMBER_OF_P...	4

Edit the Path variable to append the location of the bin folder i.e. compiler location.

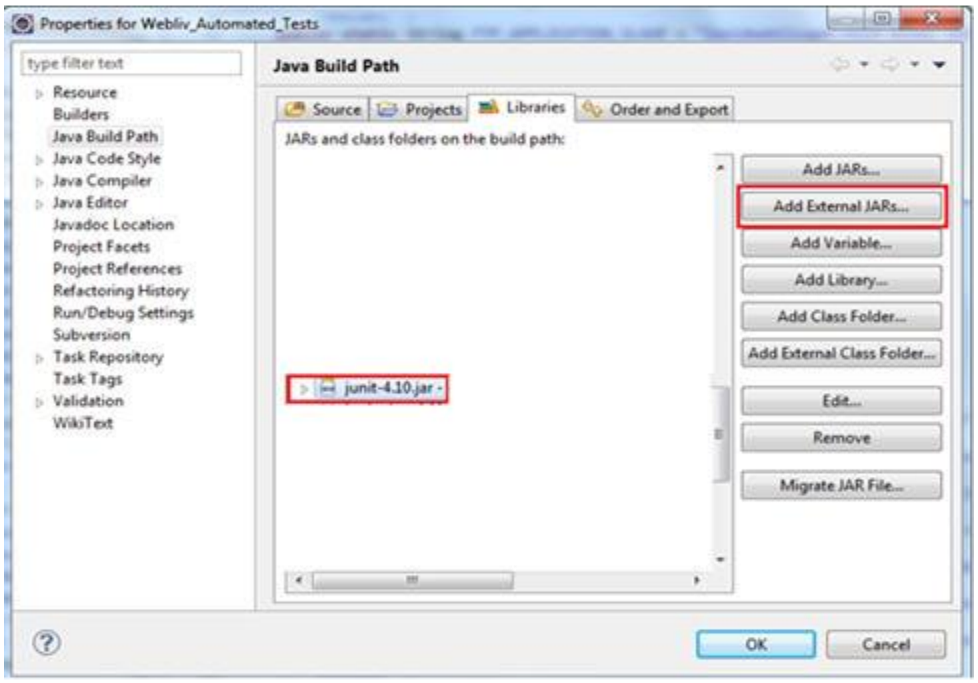
User can also verify for the successful Ant installation by typing in the “ant -version” command in the command prompt. The user would be able to see the following screen for the successful installation.



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\shrshrivastava>ant -version
Apache Ant(TM) version 1.9.2 compiled on July 8 2013
```

<u>Step 3</u>: Download and Extract Junit Jar

Download the latest version of Junit jar from “https://github.com/junit-team/junit/wiki/Download-and-Install” and configure the project’s build path in eclipse and add the jar as external library. Refer the following illustration.



Thus, no other installation is required to use Apache Ant in collaboration with Junit and Selenium WebDriver to build, execute and report the test scripts.

Note: Take a note to necessarily add “ant-junit4.jar” jar file that can be found within the library folder of the Ant’s software distribution.

Sample Build.xml

The next step is to create the project’s build file. Build file is nothing but a collection of xml elements. Worth mentioning that one build file can relate to one and only one project i.e. one build file per project or vice versa. Build file is customarily located at the project’s root/base folder but the user is leveraged to select the build’s location driven by his/her wish. Moreover the user is free to rename the build file if he/she desires.

Each of the build file must have one project and at least one target element. Refer the sample build.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="Learning_Selenium" default="junitReport"basedir=".">
    <property name="src" value="./src" />
    <property name="lib" value="./lib" />
    <property name="bin" value="./bin" />
    <property name="report" value="./report" />
    <property name="test.dir" value="./src/com/tests"/>
    <path id="Learning_Selenium.classpath">
        <pathelement location="${bin}" />
        <fileset dir="${lib}">
```

```
            <include name="**/*.jar" />
        </fileset>
    </path>
    <echo message="-----------------------------------------------------------------" />
    <echo message="--------------------Selenium Learning Tests----------------------" />
    <echo message="-----------------------------------------------------------------" />
    <target name="init" description="Delete the binary folder and create it again">
        <echo message="----------Delete the binary folder and create it again----------" />
        <delete dir="${bin}" />
        <!-- Create the time stamp -->
        <tstamp>
            <format property="lastUpdated"pattern="dd-MM-yyyy HH:mm:ss" />
        </tstamp>
        <!-- Create the build directory structure used by compile -->
        <mkdir dir="${bin}" />
    </target>
    <target name="compile" depends="init"description="Compile the source files">
        <echo message="----------Compile the source files----------" />
        <javac source="1.7" srcdir="${src}"fork="true" destdir="${bin}" includeantruntime="false"debug="true" debuglevel="lin
es,vars,source">
            <classpathrefid="Learning_Selenium.classpath" />
        </javac>
    </target>
    <target name="exec" depends="compile"description="Launch the test suite">
        <echo message="----------Launch the test suite----------" />
        <delete dir="${report}" />
        <mkdir dir="${report}" /
>
         <mkdir dir="${report}/xml" />
        <junit fork="yes"printsummary="withOutAndErr" haltonfailure="no">
            <classpathrefid="Learning_Selenium.classpath" />
            <formatter type="xml" />
            <batchtest fork="yes"todir="${report}/xml">
                <fileset dir="${src}"includes="**/com/TestSuite.java" />
            </batchtest>
        </junit>
    </target>
    <target name="junitReport" depends="exec"description="Generate the test report">
        <echo message="----------Generate the test report----------" />
        <junitreport todir="${report}">
            <fileset dir="${report}/xml">
                <include name="TEST-*.xml" />
            </fileset>
            <report format="frames"todir="${report}/html">
                <param name="TITLE"expression="Selenium_Learning_Report" />
            </report>
        </junitreport>
</target>
</project>
```

Explanation of Build.xml

The project element is fundamentally consists of 3 attributes:

<project name="Learning_Selenium" default="junitReport"basedir=".">

Each of the attribute has a "Key-Value pair" structure.

- **Name** – The value of the name attribute represents the name of the project. Thus in our case, the project's name is "Learning_Selenium".
- **Default** – The value of the default attribute represents the compulsory target for the build.xml. A build.xml file can have any number of targets. Thus this field represents the mandatory target amongst all.
- **Basedir** – Represents the root folder or base directory of the project. Under this directory, there may be several other folders like src, lib, bin etc.

<target name="init" description="Delete the binary folder and create it again">

All the tasks in the Ant build file are defined under Target elements. Each Target element corresponds to a particular task or goal. A single target can consists of multiple tasks if needed. Like I mentioned earlier, the user is credited to create more than one target within a particular build file.

In the above xml code, we have created targets for the following goals:

1. Deleting and creating directories
2. Compiling the code
3. Executing the test classes
4. Generating the test reports

<target name="exec" depends="compile" description="Launch the test suite">

Sometimes it is required to execute a particular target only when some other target is executed successfully. Take a note that the target are executed sequentially i.e. in order of sequence they are mentioned in the build file. Also I would like to mention that a particular target is executed once and only once for the current build execution. Thus, when the user is required to generate dependency between the target, he/she has to use depends attribute. The value of the "depends" attribute shall be the name of the target on which it depends. A target can depend on more than one target as well.

Built-in Tasks

Ant build file provides varieties of tasks. Few of them are discussed below:

File Tasks – File task are self explanatory.

1. <copy>
2. <concat>
3. <delete>
4. <get>
5. <mkdir>
6. <move>
7. <replace>

Compile Tasks

1. <javac> – Compiles source files within the JVM
2. <jspc> – Runs jsp compiler
3. <rmic> – Runs rmic compiler

Archive Tasks

1. <zip>, <unzip> – Creates a zipped folder
2. <jar>, <unjar> – Creates a jar file
3. <war>, <unwar> – Creates a war file for deployment

Testing Tasks

1. <junit> – Runs JUnit testing framework
2. <junitreport> – Generates the test report by converting JUnit generated XML test reports.

Property Tasks

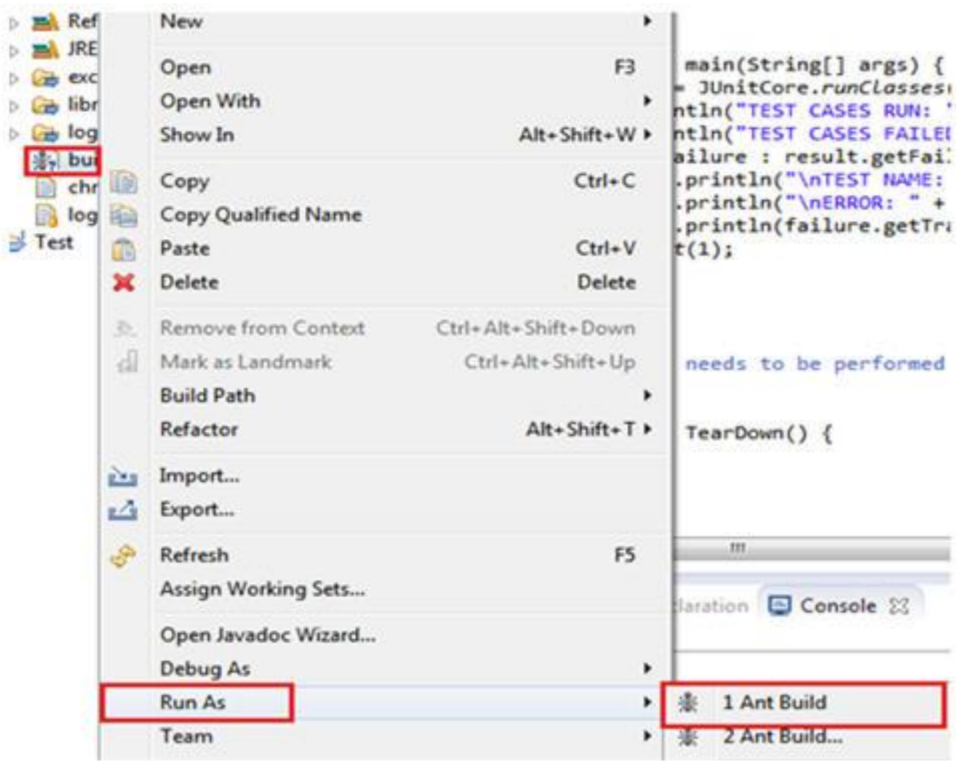
1. <dirname> – Sets the property
2. <loadfile> – Loads a file into property
3. <propertyfile> – Creates a new property file

Misc. Tasks

1. <echo> – Echoes the text message to be printed either on the console or written within an external file.
2. <javadoc> – Generates the java based documentation using javadoc tool.
3. <sql> – Establishes a JDBC connection and hits dash of SQL commands.

Execution

The easiest section is to execute the test suite with Ant. To execute the test suite with Ant, Right click on “build.xml” and select “Run As -> Ant Build” option. Thus, the option hits the execution. Refer the following figure for the same.



```
main(String[] args) {
= JUnitCore.runClasses
ntln("TEST CASES RUN: "
ntln("TEST CASES FAILE
ailure : result.getFai
.println("\nTEST NAME:
.println("\nERROR: " +
.println(failure.getTr
t(1);

needs to be performed

TearDown() {
```

After the entire execution is completed, Ant generates a test execution report for review inside the “Report” folder.

The execution can also be initiated outside the eclipse by hitting the command on the command prompt. User is expected to navigate to the directory where build.xml is kept and type “ant”.

Conclusion

In this tutorial, we laid emphasis on useful information related to Ant, its installation and various Ant tasks. Our motive was to at least introduce you with the basic conceptual picture and its importance as a tool all together with respect to testing. Hence, we discussed build.xml in detail describing the various components.

Briefing in the end, Ant is a tool that automates the software building process. Ant is not just limited to compilation of code, rather packaging, testing and a lot more can be achieved in some simple steps.

Next Tutorial #24: We will learn about Maven – a build automation tool. Maven simplifies the code handling and process of building the project. Most of the projects follow maven structure. We will learn how to use Maven and Maven project setup for Selenium.

Use of Maven Build Automation Tool and Maven Project Setup for Selenium – Selenium Tutorial #24

In our last Selenium tutorial we learned a build tool named as "Apache Ant". We also broadly discussed its applicability and importance besides the practical approach.

In this Selenium Testing tutorial we will learn **Maven – a build automation tool** which is distributed under Apache Software Foundation. It is mainly used for java projects. It makes build consistent with other project. Maven is also used to manage the dependencies. For example if you are using selenium version 2.35 and any later point of time you have to use some other version, then same can be managed easily by Maven. You will find more examples of this later in this chapter. It works very effectively when there is huge number of Jar files with different versions.

What is a build tool?

Build tool is used to setup everything which is required to run your java code independently. This can be applied to your entire java project. It generates source code, compiling code, packaging code to a jar etc. Maven provides a common platform to perform these activities which makes programmer's life easier while handling huge project.

Maven provides *pom.xml* which is the core to any project. This is the configuration file where all required information's are kept. Many of the IDEs (Integrated Development Environments) are available which makes it easy to use. IDEs are available for tools like Eclipse , NetBeans, IntelliJ etc.

Maven stores all project jars. Library jar are in place called repository which could be central, local or remote repository. Maven downloads the dependency jar from central repository. Most of the commonly used libraries are available in http://repo1.maven.org/maven2/.

Downloaded libraries are stored in local repository called m2. Maven uses the libraries available in m2 folder and if any new dependency added then maven downloads from central repository to local repository. If libraries are not available in central repository then maven looks for remote repository. User has to configure the remote repository in *pom.xml* to download from remote repository.

Below is the example of configuring a remote repository to *pom.xml* file. Provide id and url of the repository where libraries are stored.

```
<repositories>
    <repository>
       <id>libraryId</id>
       <url>http://comanyrepositryId</url>
    </repository>
</repositories>
```

General Phrases used in Maven:

- **groupId**: Generally groupId refers to domain id. For best practices company name is used as groupId. It identifies the project uniquely.
- **artifactId**: It is basically the name of the Jar without version.
- **version**: This tag is used to create a version of the project.
- **Local repository**: Maven downloads all the required dependencies and stores in local repository called m2. More details regarding the same would be shared in the next topic.

Build Life Cycle:

Basic maven phases are used as below.

- **clean**: deletes all artifacts and targets which are created already.
- **compile**: used to compile the source code of the project.
- **test**: test the compiled code and these tests do not require to be packaged or deployed.
- **package**: package is used to convert your project into a jar or war etc.
- **install**: install the package into local repository for use of other project.

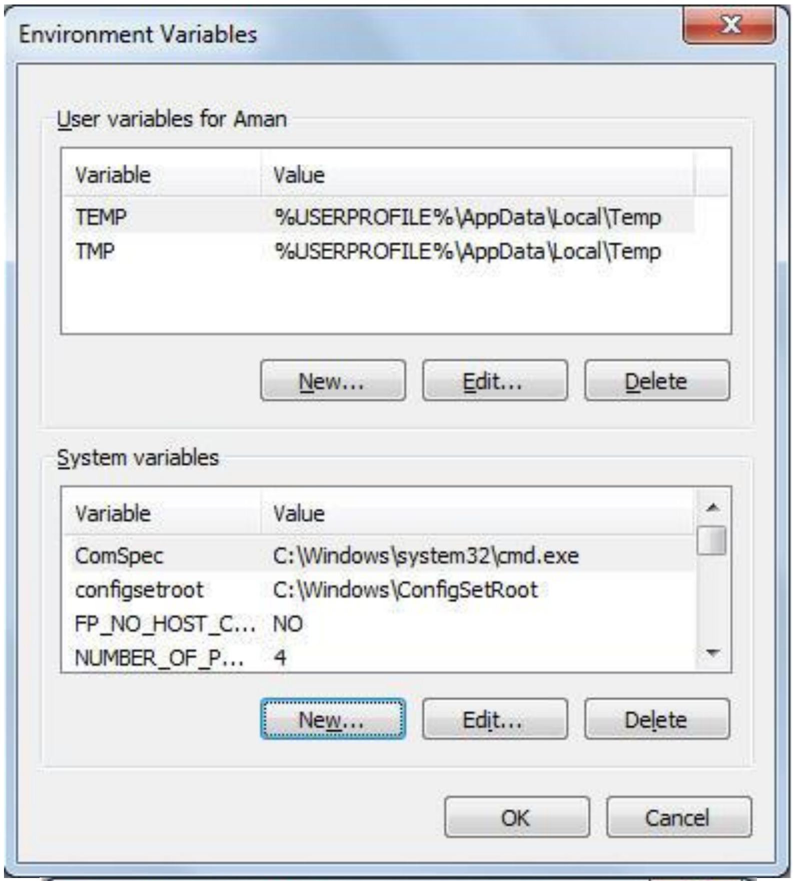
Maven Setup:

Step 1: To setup Maven, download the maven’s latest version form Apache depending upon different OS.

Step 2: Unzip the folder and save it on the local disk.

Step 3: Create environment variable for MAVEN_HOME. Follow the below step:

Navigate to System Properties ->Advanced System Setting->Environment Variable ->System Variable ->New ->Add path of Maven folder

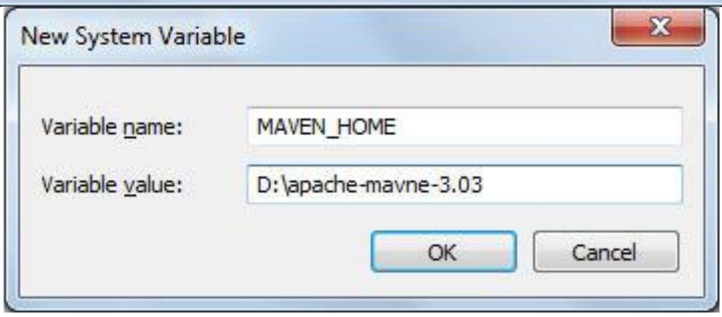


User variables for Aman

Variable	Value
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp

System variables

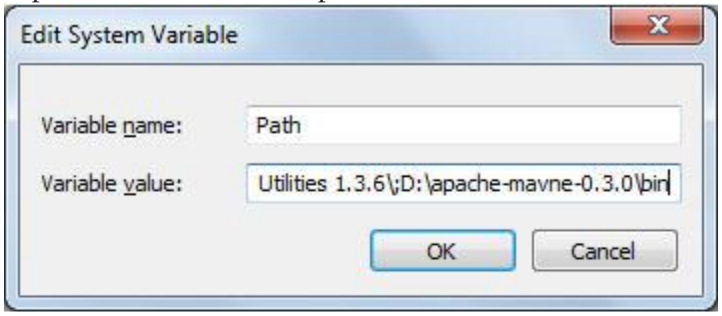
Variable	Value
ComSpec	C:\Windows\system32\cmd.exe
configsetroot	C:\Windows\ConfigSetRoot
FP_NO_HOST_C...	NO
NUMBER_OF_P...	4



Variable name: MAVEN_HOME

Variable value: D:\apache-mavne-3.03

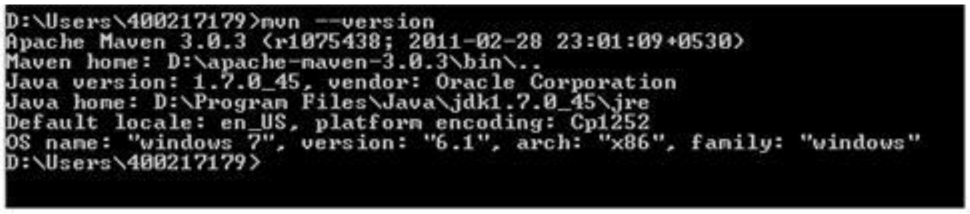
Step 4: Edit path variable and provide the bin folder path.



Variable name: Path

Variable value: Utilities 1.3.6\;D:\apache-mavne-0.3.0\bin

Step 5: Now verify the maven installation using command prompt and don’t forget to setup JAVA_HOME Use mvn –version to verify maven version and output comes like below.

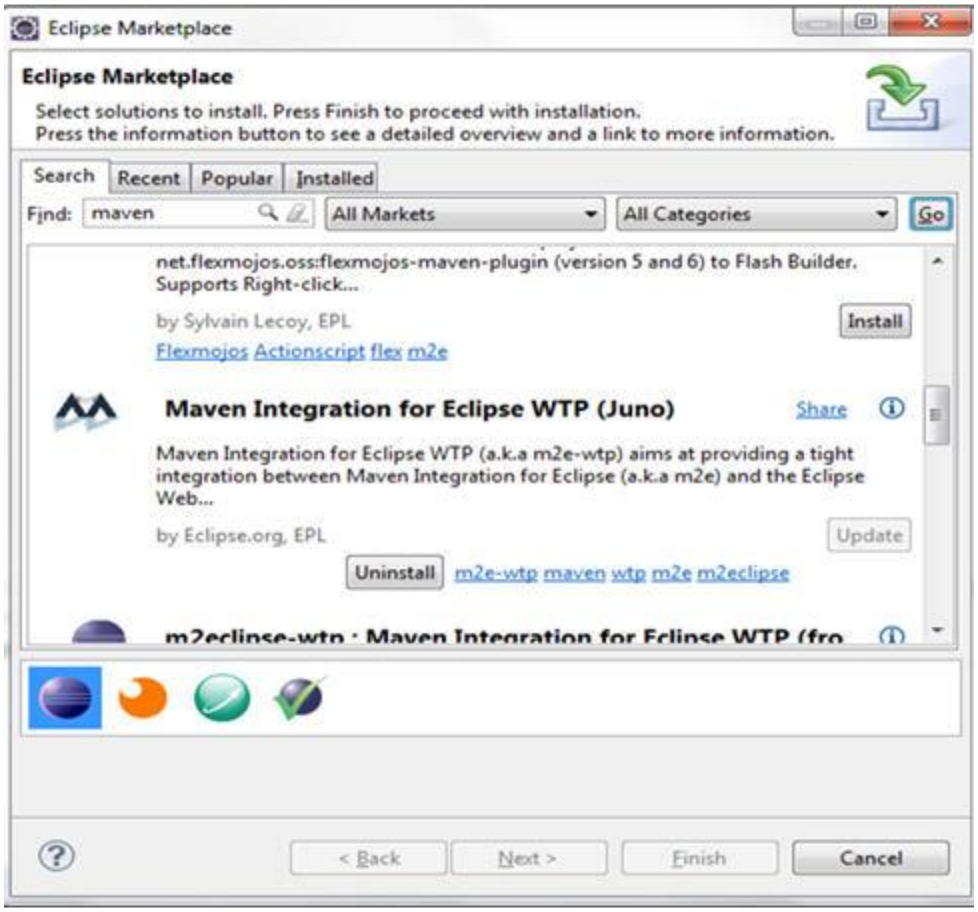


```
D:\Users\400217179>mvn --version
Apache Maven 3.0.3 (r1075438; 2011-02-28 23:01:09+0530)
Maven home: D:\apache-maven-3.0.3\bin\..
Java version: 1.7.0_45, vendor: Oracle Corporation
Java home: D:\Program Files\Java\jdk1.7.0_45\jre
Default locale: en_US, platform encoding: Cp1252
OS name: "windows 7", version: "6.1", arch: "x86", family: "windows"
D:\Users\400217179>
```

Install maven IDE in Eclipse:

Maven provides IDE to integrate with eclipse. I am using eclipse Juno here.

Navigate to Help->Eclipse Marketplace-> Search maven ->Maven Integration for Eclipse ->INSTALL



Eclipse Marketplace

Eclipse Marketplace

Select solutions to install. Press Finish to proceed with installation.
Press the information button to see a detailed overview and a link to more information.

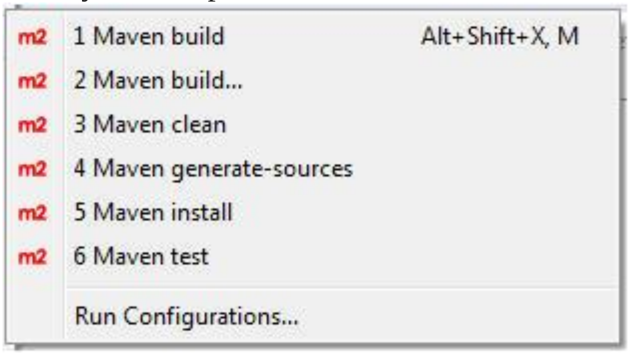
net.flexmojos.oss:flexmojos-maven-plugin (version 5 and 6) to Flash Builder. Supports Right-click...
by Sylvain Lecoy, EPL

Maven Integration for Eclipse WTP (Juno)

Maven Integration for Eclipse WTP (a.k.a m2e-wtp) aims at providing a tight integration between Maven Integration for Eclipse (a.k.a m2e) and the Eclipse Web...
by Eclipse.org, EPL
Share
Update
Uninstall
m2e-wtp maven wtp m2e m2eclipse
m2eclipse-wtp : Maven Integration for Eclipse WTP (fro
< Back
Next >
Finish
Cancel

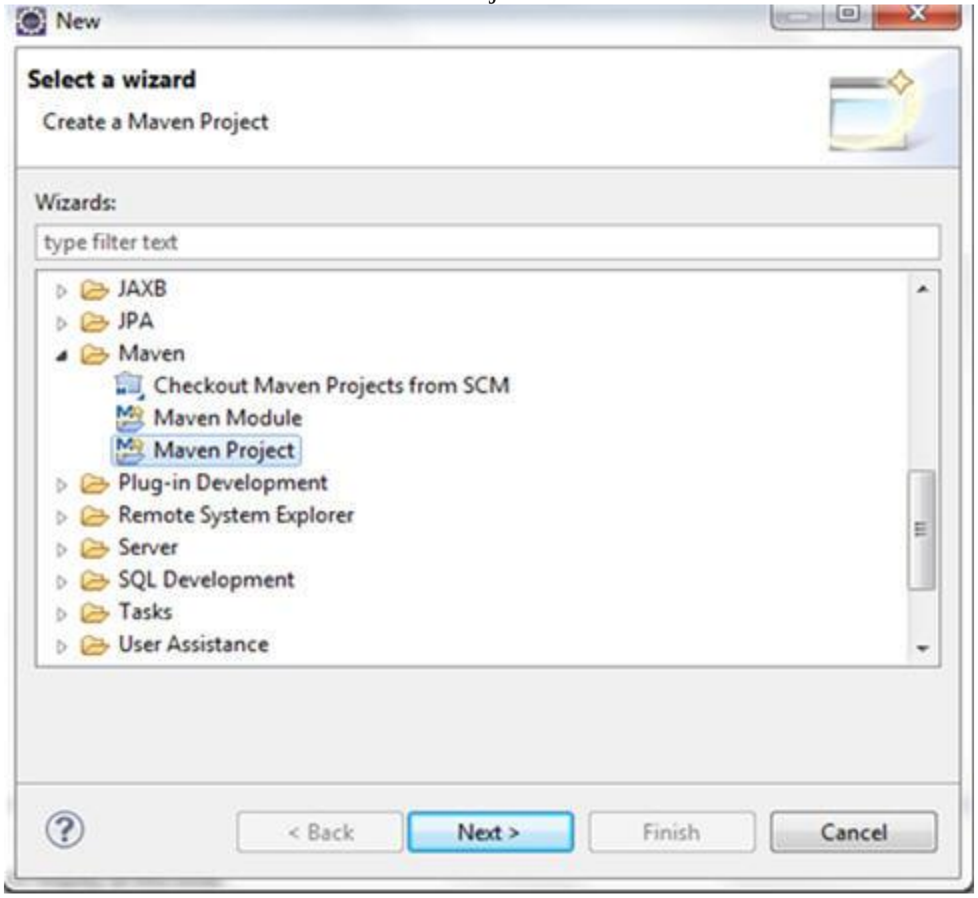
After installation you have to restart eclipse.

Then right click on *pom.xml* and verify all the options are available like below.

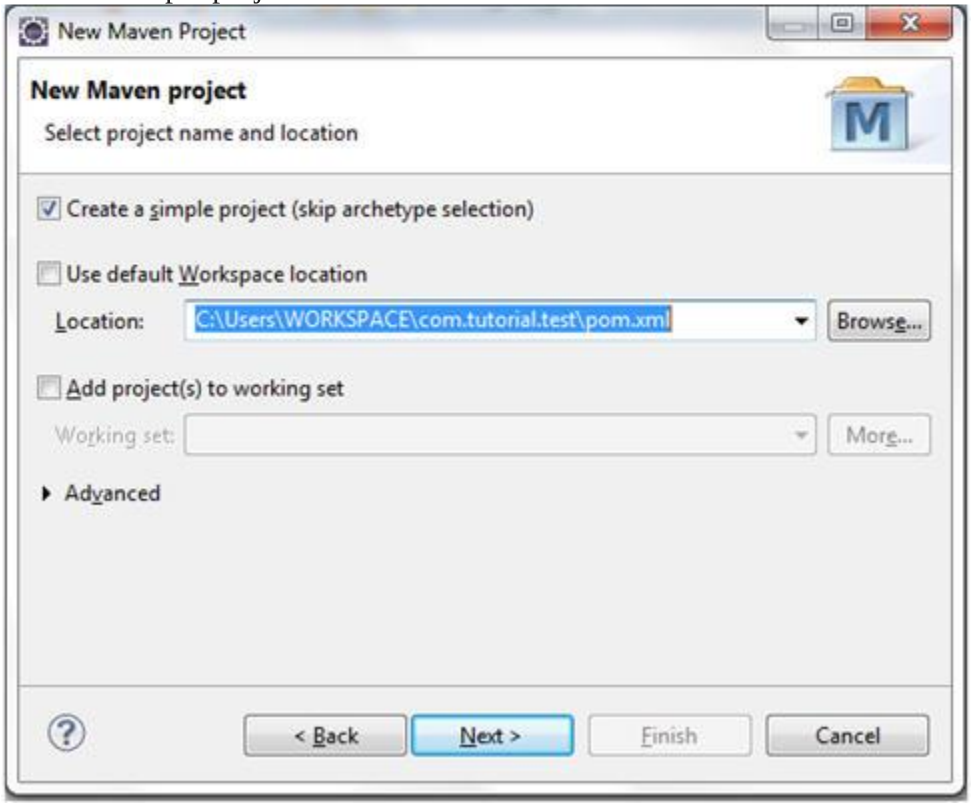


Create Maven project:

Step 1: Navigate to File- new-others-Maven-Maven Project-Click Next



Step 2: Check the Create a simple project and click Next



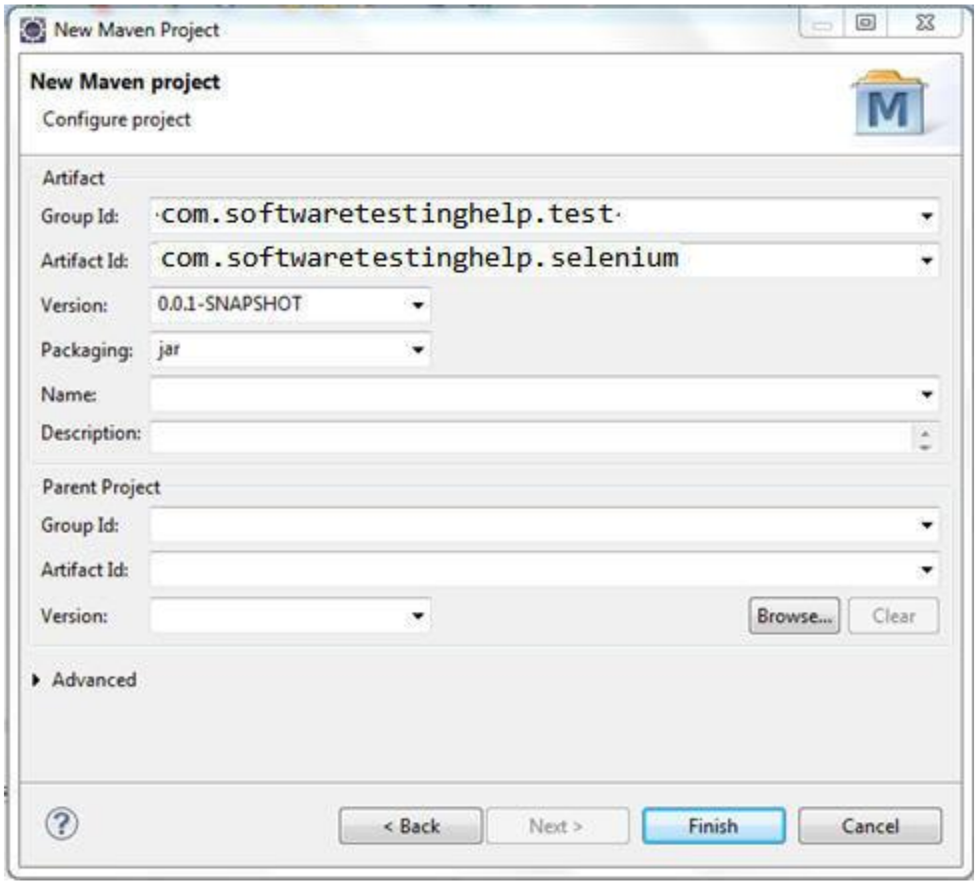
☑ Create a simple project (skip archetype selection)

☐ Use default Workspace location

Location: C:\Users\WORKSPACE\com.tutorial.test\pom.xml

Browse...

Step 3: Provide Group Id and Artifact Id .You can change the version of Jar as per your wish. Here I am using default name. Click Finish.



New Maven Project

New Maven project

Configure project

Artifact

Group Id: com.softwaretestinghelp.test

Artifact Id: com.softwaretestinghelp.selenium

Version: 0.0.1-SNAPSHOT

Packaging: jar

Name:

Description:

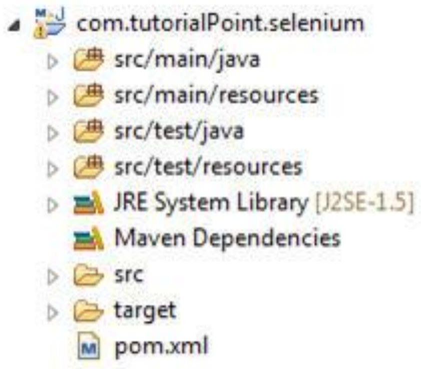
Parent Project

Group Id:

Artifact Id:

Version:

Step 4: After finish you will find the project structure is created like below. *pom.xml* is created which is used to download all dependencies.



com.tutorialPoint.selenium
src/main/java
src/main/resources
src/test/java
src/test/resources
JRE System Library [J2SE-1.5]
Maven Dependencies
src
target
pom.xml

pom.xml file looks like below:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
 <modelVersion>4.0.0</modelVersion>
<groupId>com.softwaretestinghelp.test</groupId>
<artifactId>com.softwaretestinghelp.selenium</artifactId>
<version>0.0.1-SNAPSHOT</version>
```

```
</project>
```

Step 5: Add dependencies for Selenium.

All selenium Maven artifacts are available in below central repository

http://repo1.maven.org/maven2/org/seleniumhq/selenium/

Add following dependencies in *pom.xml* for selenium

```
<dependency>
     <groupId>org.seleniumhq.selenium</groupId>
     <artifactId>selenium-java</artifactId>
     <version>2.41.0</version>
 </dependency>
```

Similarly, following is the dependency for Junit :

```
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.4</version>
 </dependency>
```

If you want to add other third party jar then add those dependencies in *pom.xml*

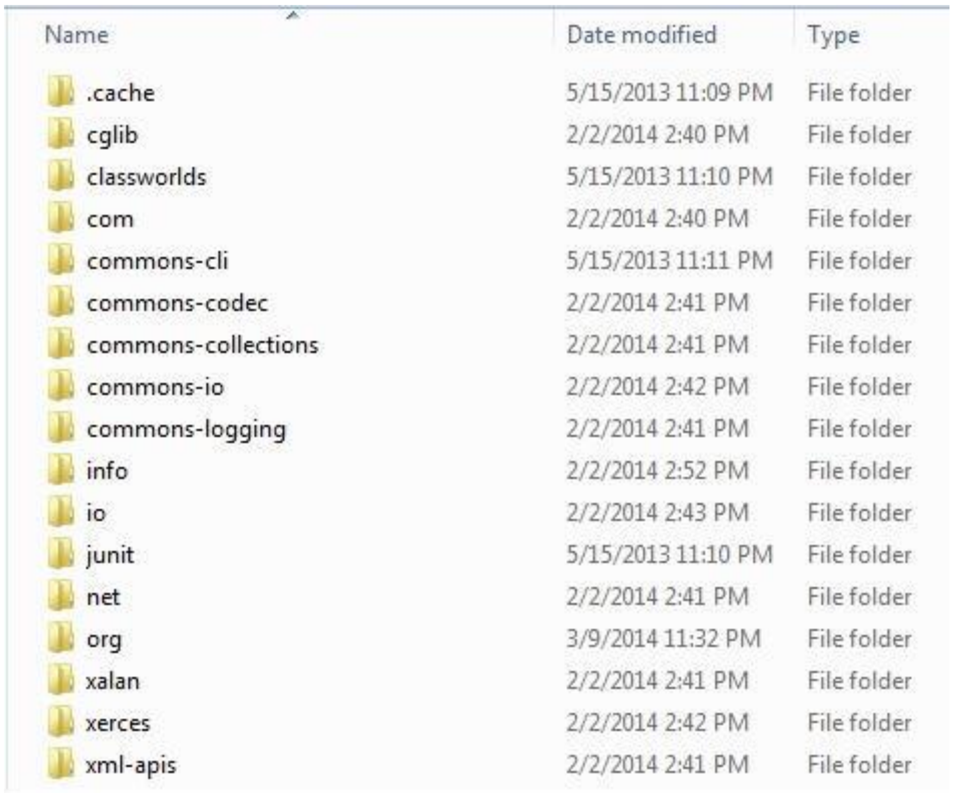
Step 6: Final *pom.xml* will be like below:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
 <modelVersion>4.0.0</modelVersion>
<groupId>com.softwaretestinghelp.test</groupId> <artifactId>com.softwaretestinghelp.selenium</artifactId>
<version>0.0.1-SNAPSHOT</version>
<dependencies>
<dependency>
     <groupId>org.seleniumhq.selenium</groupId>
     <artifactId>selenium-java</artifactId>
      <version>2.41.0</version>
  </dependency>
  </dependencies>
</project>
```

Step 7: Maven will download all the dependency jars in to local repository called .m2.

M2 folder is basically inside Users->username->m2

All the jars will be placed in a folder called repository which is inside .m2 folder. Maven will create separate folders for different version and different group id.

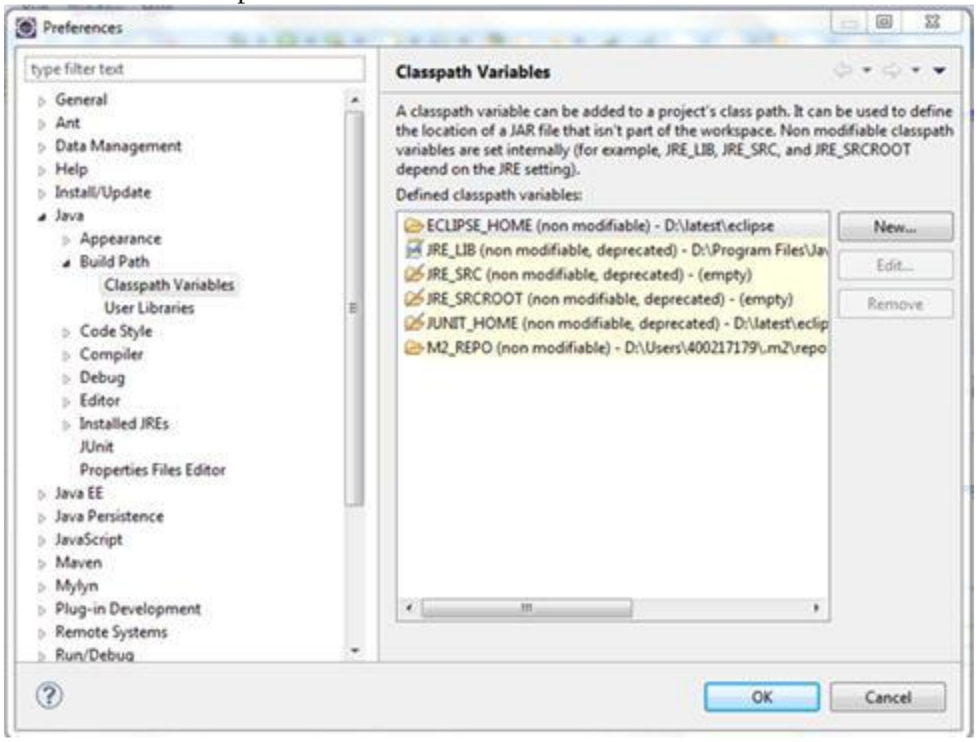


Name	Date modified	Type
.cache	5/15/2013 11:09 PM	File folder
cglib	2/2/2014 2:40 PM	File folder
classworlds	5/15/2013 11:10 PM	File folder
com	2/2/2014 2:40 PM	File folder
commons-cli	5/15/2013 11:11 PM	File folder
commons-codec	2/2/2014 2:41 PM	File folder
commons-collections	2/2/2014 2:41 PM	File folder
commons-io	2/2/2014 2:42 PM	File folder
commons-logging	2/2/2014 2:41 PM	File folder
info	2/2/2014 2:52 PM	File folder
io	2/2/2014 2:43 PM	File folder
junit	5/15/2013 11:10 PM	File folder
net	2/2/2014 2:41 PM	File folder
org	3/9/2014 11:32 PM	File folder
xalan	2/2/2014 2:41 PM	File folder
xerces	2/2/2014 2:42 PM	File folder
xml-apis	2/2/2014 2:41 PM	File folder

Step 8: If m2 folder does not populate in Maven dependencies, then you can populate those jars manually.

– Eclipse Windows ->Preference

– Navigate Java->Build Path->Classpath Variables



Classpath Variables

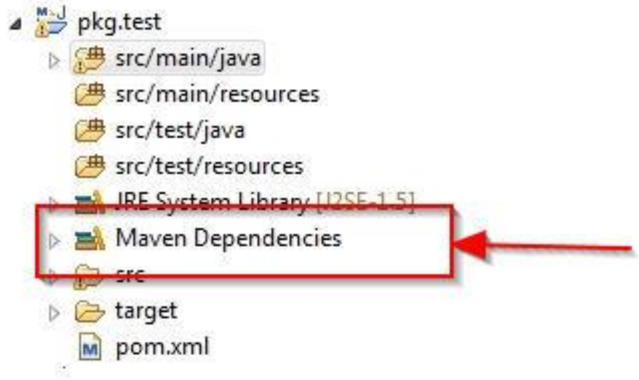
A classpath variable can be added to a project's class path. It can be used to define the location of a JAR file that isn't part of the workspace. Non modifiable classpath variables are set internally (for example, JRE_LIB, JRE_SRC, and JRE_SRCROOT depend on the JRE setting).

Defined classpath variables:

```
ECLIPSE_HOME (non modifiable) - D:\latest\eclipse
JRE_LIB (non modifiable, deprecated) - D:\Program Files\Ja
JRE_SRC (non modifiable, deprecated) - (empty)
JRE_SRCROOT (non modifiable, deprecated) - (empty)
JUNIT_HOME (non modifiable, deprecated) - D:\latest\eclip
M2_REPO (non modifiable) - D:\Users\400217179\.m2\repo
```

– Click New Button ->Define M2_REPO and provide the path of m2 folder.

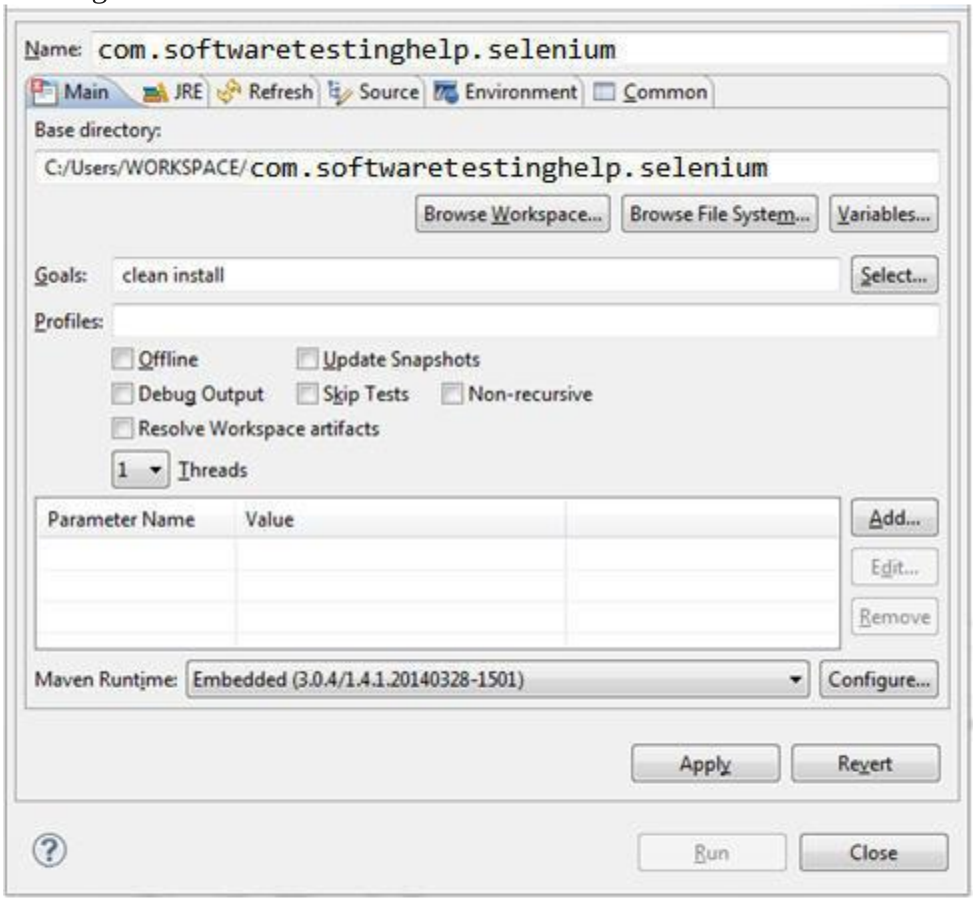
Step 9: Upon successful setup you will find Maven Dependencies folder like below which will have the required dependency jar for the project



Build the Project:

Project can be built by both using IDE and command prompt.

Using IDE you have to right click on POM-*Run As-Maven* Build


Name: com.softwaretestinghelp.selenium

Base directory: C:/Users/WORKSPACE/com.softwaretestinghelp.selenium

Goals: clean install

Profiles:

Maven Runtime: Embedded (3.0.4/1.4.1.20140328-1501)

Enter goals like clean install etc. and click Run.

Same can be done using command prompt. Navigate to project folder where *pom.xml* lies.

And use below commands to clean, compile and install

For clean: mvn clean

For compile: mvn compile

For Install: mvn install

Below is the info which is displayed when you clean any project and shows “BUILD SUCCESS”.

```
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building com.softwaretestinghelp.0.0.1-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
 [INFO] --- maven-clean-plugin:2.4.1:clean (default-clean) @ com.softwaretestinghelp ---[INFO] Deleting
C:\Users\rshwus\WORKSPACE\com.softwaretestinghelp\target
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
                  [INFO] ------------------------------------------------------------------------
[INFO] Total time: 0.702s
[INFO] Finished at: Sat May 24 18:58:22 IST 2014
[INFO] Final Memory: 2M/15M
[INFO] ------------------------------------------------------------------------
```

Conclusion:

Maven simplifies the code handling and process of building the project. Most of the projects follow maven structure.

Download all dependencies provided the dependencies are available in maven central repository. If any of the dependency is not available in maven central repository then you have to add repository path in pom.xml explicitly.

There are many other build tools available in like ant. But it is better to use maven while dealing with different versions and different dependencies. Maven even can manage the dependencies of dependencies. Other tools may not provide such flexibility like maven. Please post your queries anything related to maven here.

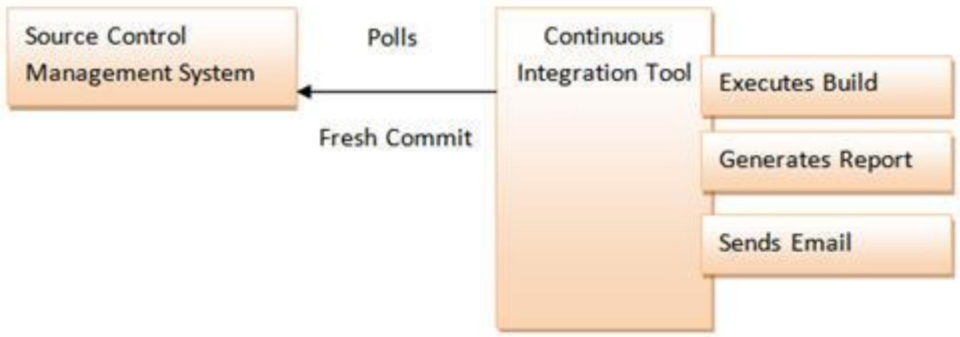
Next Tutorial #25: In the upcoming tutorial, we would discuss about **continuous integration tool known as Hudson**. We would study about its importance, role and benefits into Test Automation Framework. We would look at the Hudson straight from the beginning, from its installation to its working.

Hudson – Importance and Benefits of this Continuous Integration Tool – Selenium Tutorial #25

In the last two tutorials, we discussed about the two most important build tools – ANT and Maven. We discussed about their significance and practical importance.
In the current Selenium online training tutorial, we would discuss about a **continuous integration tool known as Hudson**. We would study about its importance and benefits that we get out of any continuous integration tool. We would look at the Hudson straight from the beginning, from its installation to its advance settings.

Continuous Integration
Many a times, we end up working in a project where a large bunch of developers and testers are working together on different modules. Developers and Testers work on their modules thereby developing executables. These work products are then integrated at regular intervals. Thus, every time we create a development code, it needs to be integrated, tested and built to ensure that the code developed doesn't break or introduces errors or defects.

This process of building and testing the development work integrated at regular intervals is known as **Continuous Integration (CI)**. Continuous Integration lets you identify and address the defects or errors as soon as possible in the development lifecycle i.e. closer to the time, they were introduced.



Continuous Integration system builds and tests the application as soon as the fresh/changed code is committed to the Source Control Management system acronym as SCM. With its great benefits and impact over the industries, it has become an integral part of Software Development life cycle and is mandatorily practiced.

Hudson – Continuous Integration Tool
Continuous Integration can be performed automatically. Hudson is one of the popularly known tools to perform Continuous Integration. Hudson is a Java based open source Continuous Integration tool. Like any other Continuous Integration tool, Hudson provides the teams to trigger builds and tests with any change in the Source Control Management System.

Hudson supports a large range of tools and plugins.

Hudson:

- Supports SCM tools like CVS, Subversion (SVN), Git etc.
- Is capable of building ANT based projects, Maven based projects etc.
- Is capable of executing shell scripts and Windows batch commands
- Is capable of Sending reports, notifications etc via Email, SMS, Skype etc.

Hudson Installation

Pre-requisites

To be able to use Hudson, we need the following things to be in place before we get started:

- Source Code Repository (SVN/Git/CVS etc.)
- Build Script (Ant/Maven etc.)

Installation

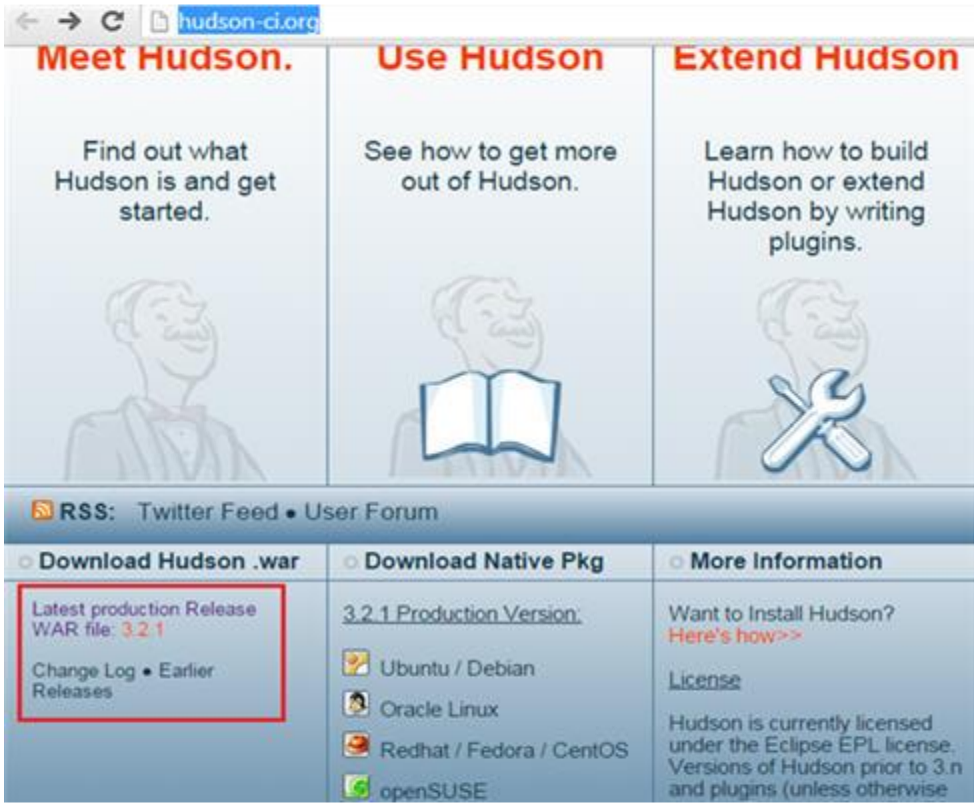
Hudson can be easily installed on variety of environments. Hudson can be installed on both the Linux machine and the windows machine. It is also distributed as a package specific to the OS type for different Linux flavors thereby making the installation a few minute's task. Hudson can be run as a standalone application or within the Servlet Container. In this tutorial, we would explain the Hudson Installation on Windows machine. There are two distinct approaches to install Hudson.

- Using WAR file
- Using Native Package

Native Packages are available for Ubuntu/Debian, Oracle Linux, Redhat/Fedora/CentOS and openSUSE.

For this tutorial, we would discuss installation by WAR file. Let us discuss the entire process step by step.

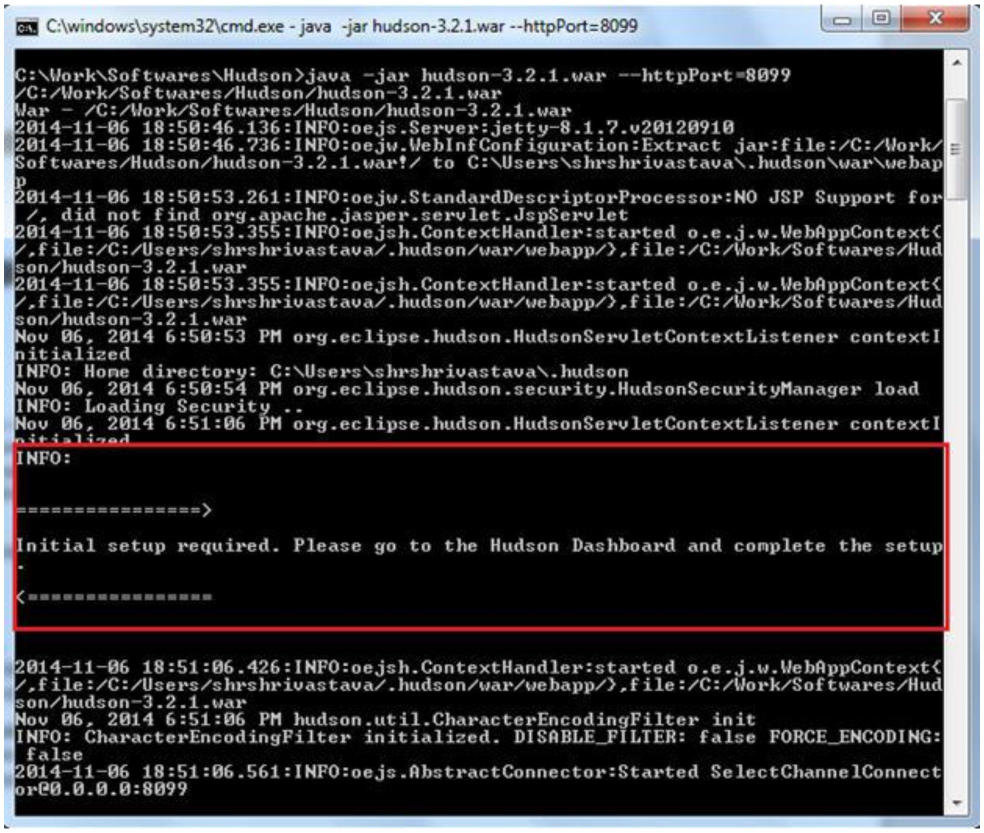
Step 1: Download the Hudson WAR file from the Hudson's official website – "http://hudson-ci.org/". Keep the war file in the desired location in the local file system. This WAR file can be started directly via command prompt or can be used in Servlet Container. The WAR is an executable file that has a Servlet Container embedded in itself.


hudson-ci.org
Meet Hudson.
Find out what Hudson is and get started.
Use Hudson
See how to get more out of Hudson.
Extend Hudson
Learn how to build Hudson or extend Hudson by writing plugins.
RSS: Twitter Feed • User Forum
Download Hudson .war
Latest production Release WAR file: 3.2.1
Change Log • Earlier Releases
Download Native Pkg
3.2.1 Production Version:
Ubuntu / Debian
Oracle Linux
Redhat / Fedora / CentOS
openSUSE
More Information
Want to Install Hudson? Here's how>>
License
Hudson is currently licensed under the Eclipse EPL license. Versions of Hudson prior to 3.n and plugins (unless otherwise

Step 2: The next step is to initialize the Hudson web user interface. For this, we need to open command prompt and go to the folder where Hudson war is kept.

- Type java -jar hudson-3.0.1.war –httpPort=8099

The above command would show that the Initial setup is required to be done on Hudson Dashboard. Refer the below screen.



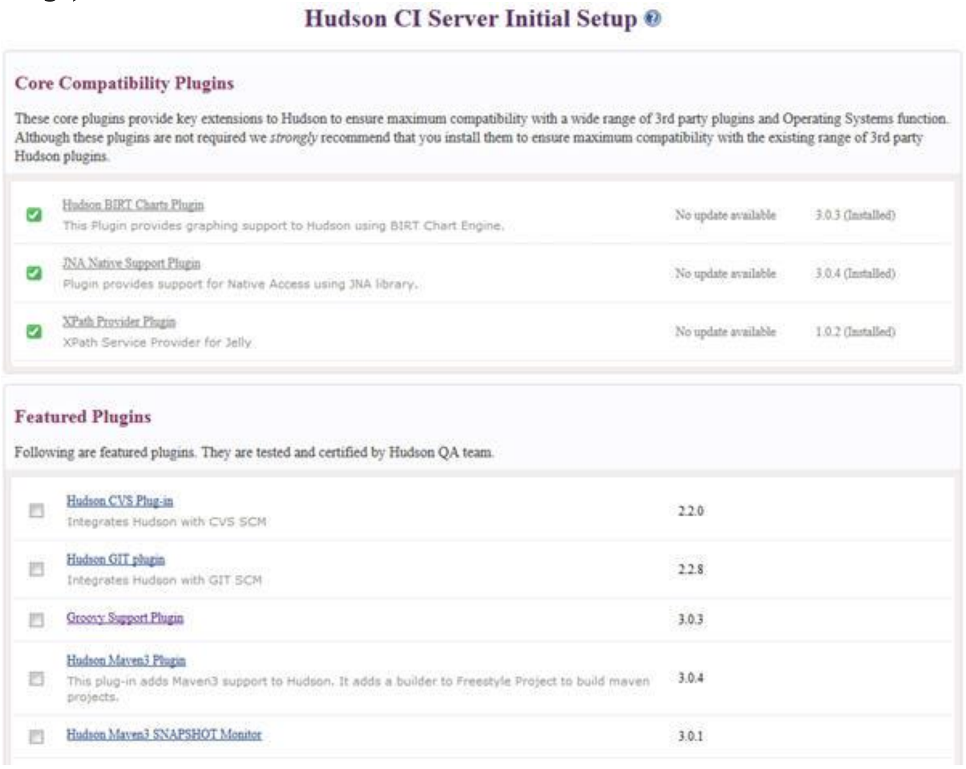
```
C:\windows\system32\cmd.exe - java  -jar hudson-3.2.1.war --httpPort=8099

C:\Work\Softwares\Hudson>java -jar hudson-3.2.1.war --httpPort=8099
/C:/Work/Softwares/Hudson/hudson-3.2.1.war
War - /C:/Work/Softwares/Hudson/hudson-3.2.1.war
2014-11-06 18:50:46.136:INFO:oejs.Server:jetty-8.1.7.v20120910
2014-11-06 18:50:46.736:INFO:oejw.WebInfConfiguration:Extract jar:file:/C:/Work/
Softwares/Hudson/hudson-3.2.1.war!/ to C:\Users\shrshrivastava\.hudson\war\webap
p
2014-11-06 18:50:53.261:INFO:oejw.StandardDescriptorProcessor:NO JSP Support for
 /, did not find org.apache.jasper.servlet.JspServlet
2014-11-06 18:50:53.355:INFO:oejsh.ContextHandler:started o.e.j.w.WebAppContext{
/,file:/C:/Users/shrshrivastava/.hudson/war/webapp/},file:/C:/Work/Softwares/Hud
son/hudson-3.2.1.war
2014-11-06 18:50:53.355:INFO:oejsh.ContextHandler:started o.e.j.w.WebAppContext{
/,file:/C:/Users/shrshrivastava/.hudson/war/webapp/},file:/C:/Work/Softwares/Hud
son/hudson-3.2.1.war
Nov 06, 2014 6:50:53 PM org.eclipse.hudson.HudsonServletContextListener contextI
nitialized
INFO: Home directory: C:\Users\shrshrivastava\.hudson
Nov 06, 2014 6:50:54 PM org.eclipse.hudson.security.HudsonSecurityManager load
INFO: Loading Security ..
Nov 06, 2014 6:51:06 PM org.eclipse.hudson.HudsonServletContextListener contextI
nitialized
INFO:


================>

Initial setup required. Please go to the Hudson Dashboard and complete the setup
.

<================


2014-11-06 18:51:06.426:INFO:oejsh.ContextHandler:started o.e.j.w.WebAppContext{
/,file:/C:/Users/shrshrivastava/.hudson/war/webapp/},file:/C:/Work/Softwares/Hud
son/hudson-3.2.1.war
Nov 06, 2014 6:51:06 PM hudson.util.CharacterEncodingFilter init
INFO: CharacterEncodingFilter initialized. DISABLE_FILTER: false FORCE_ENCODING:
 false
2014-11-06 18:51:06.561:INFO:oejs.AbstractConnector:Started SelectChannelConnect
or@0.0.0.0:8099
```

Note: It is advisable to start Hudson as a service on Windows or Linux machine.

Step 3: To be able to access the Hudson window, open your browser and launch Hudson.

- Type "http://localhost:8099/" – This will open hudson window.

(Click to enlarge image)



Hudson CI Server Initial Setup

Core Compatibility Plugins

These core plugins provide key extensions to Hudson to ensure maximum compatibility with a wide range of 3rd party plugins and Operating Systems function. Although these plugins are not required we *strongly* recommend that you install them to ensure maximum compatibility with the existing range of 3rd party Hudson plugins.

✓	Hudson BIRT Charts Plugin This Plugin provides graphing support to Hudson using BIRT Chart Engine.	No update available	3.0.3 (Installed)
✓	JNA Native Support Plugin Plugin provides support for Native Access using JNA library.	No update available	3.0.4 (Installed)
✓	XPath Provider Plugin XPath Service Provider for Jelly	No update available	1.0.2 (Installed)

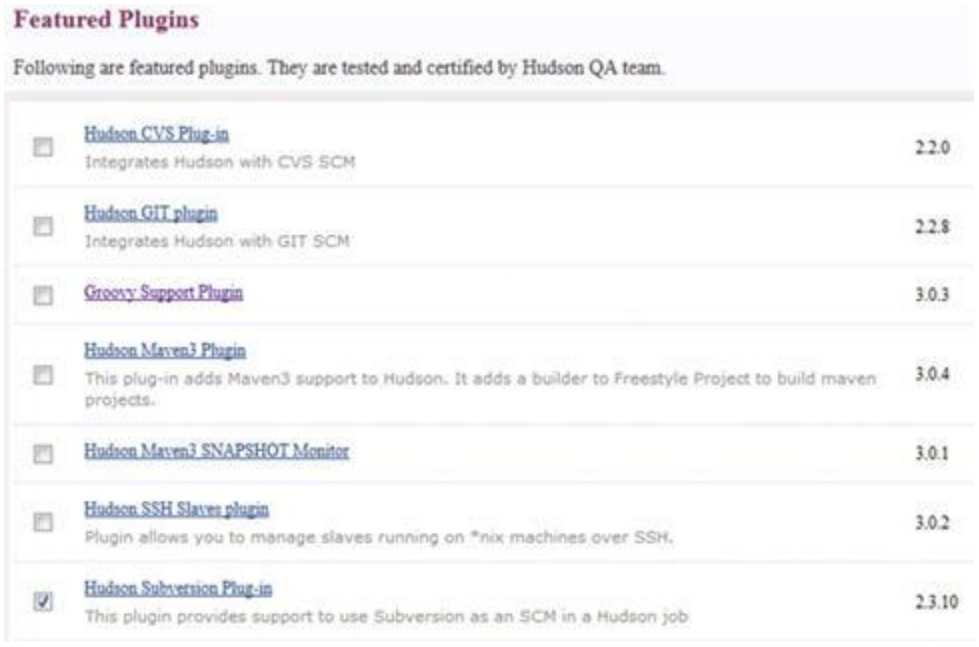
Featured Plugins

Following are featured plugins. They are tested and certified by Hudson QA team.

☐	Hudson CVS Plug-in Integrates Hudson with CVS SCM	2.2.0
☐	Hudson GIT plugin Integrates Hudson with GIT SCM	2.2.8
☐	Groovy Support Plugin	3.0.3
☐	Hudson Maven3 Plugin This plug-in adds Maven3 support to Hudson. It adds a builder to Freestyle Project to build maven projects.	3.0.4
☐	Hudson Maven3 SNAPSHOT Monitor	3.0.1

Step 4: Select the desired plugins and click on the Finish button. Please be patient as it is likely to take a few minutes to install all the plugins.

Note: There are several options available to provide support for SCM. Check mark the SCM option, you wish to use.

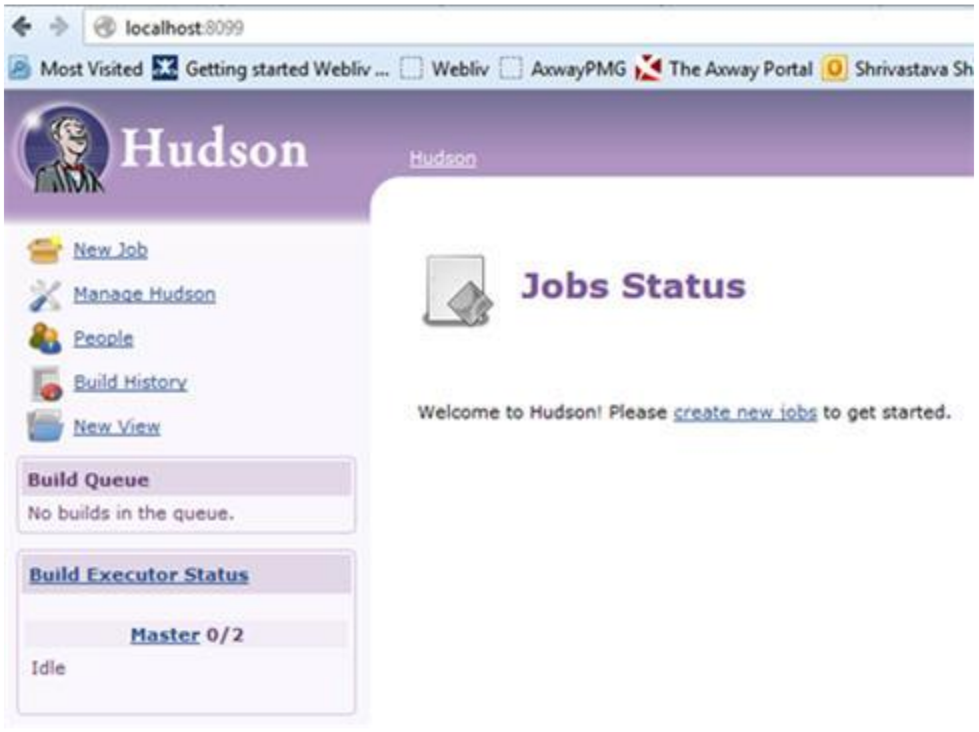


Featured Plugins

Following are featured plugins. They are tested and certified by Hudson QA team.

☐	Hudson CVS Plug-in Integrates Hudson with CVS SCM	2.2.0
☐	Hudson GIT plugin Integrates Hudson with GIT SCM	2.2.8
☐	Groovy Support Plugin	3.0.3
☐	Hudson Maven3 Plugin This plug-in adds Maven3 support to Hudson. It adds a builder to Freestyle Project to build maven projects.	3.0.4
☐	Hudson Maven3 SNAPSHOT Monitor	3.0.1
☐	Hudson SSH Slaves plugin Plugin allows you to manage slaves running on *nix machines over SSH.	3.0.2
✓	Hudson Subversion Plug-in This plugin provides support to use Subversion as an SCM in a Hudson job	2.3.10

Once, all the plugins have been installed, user can view the Hudson Dashboard.



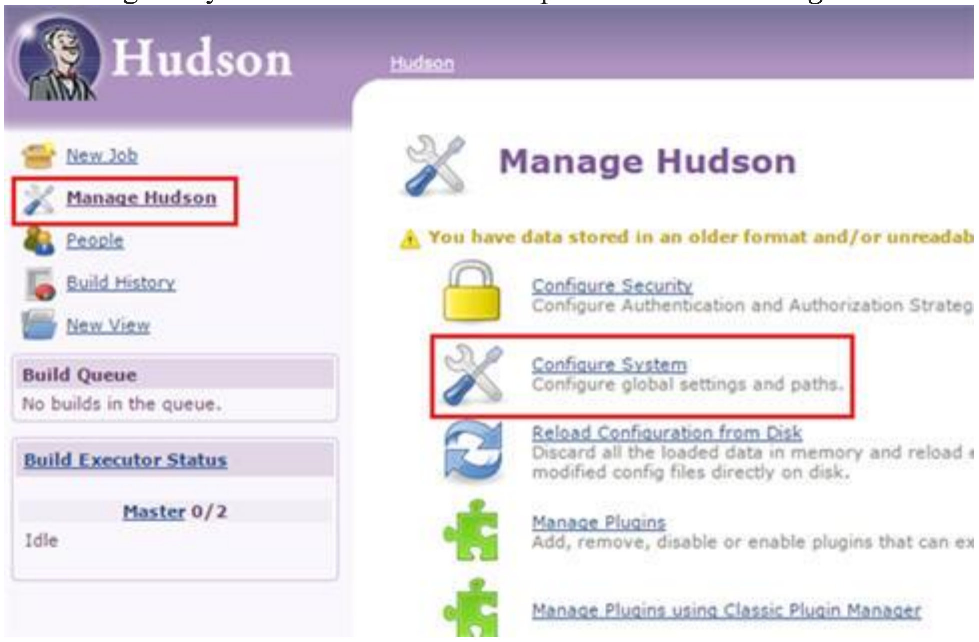
No builds in the queue.

Welcome to Hudson! Please create new jobs to get started.

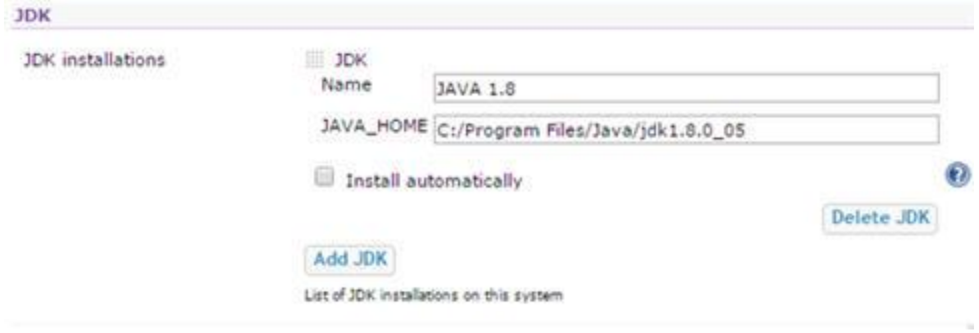
Hudson Configuration

Now that the Hudson Dashboard is ready, the next step is to configure the Hudson. Let us again discuss the entire process in steps:

Step 1: To configure the Hudson, click on the "Manage Hudson" link displayed in the left menu.
Step 2: Click on the "Configure System" link in the next step. Refer the following screenshot.


Hudson
Hudson
New Job
Manage Hudson
People
Build History
New View
Build Queue
No builds in the queue.
Build Executor Status
Master 0/2
Idle
Manage Hudson
You have data stored in an older format and/or unreadab
Configure Security
Configure Authentication and Authorization Strateg
Configure System
Configure global settings and paths.
Reload Configuration from Disk
Discard all the loaded data in memory and reload
modified config files directly on disk.
Manage Plugins
Add, remove, disable or enable plugins that can ex
Manage Plugins using Classic Plugin Manager

Step 3: As soon as you click on the Configure system link, numerous sections for connection parameters would be should. Add an entry to JDK as shown in the following figure. User needs to provide the name of the JDK installation and the location where java is installed. More than one Java instances can be added.

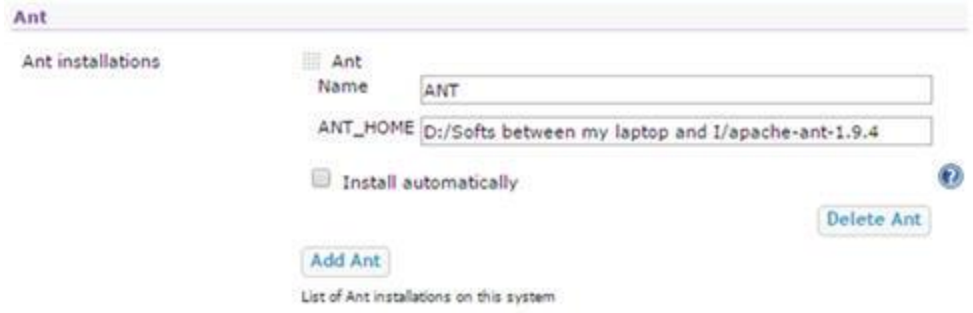

JDK

JDK installations

JDK
Name: JAVA 1.8
JAVA_HOME: C:/Program Files/Java/jdk1.8.0_05
Install automatically
Delete JDK
Add JDK
List of JDK installations on this system

User can also install JDK automatically by checking a "Install automatically" checkbox.

Step 4: In the next step, add an entry to Ant as shown in the following figure. User needs to provide the name of the Ant installation and the location where Ant is installed locally.


Ant

Ant installations

Ant
Name: ANT
ANT_HOME: D:/Softs between my laptop and I/apache-ant-1.9.4
Install automatically
Delete Ant
Add Ant
List of Ant installations on this system

Like JDK and Ant, user can configure other connection parameters.

Note: Always remember to uncheck the “Install automatically” checkbox. The checkbox should be selected in case if you wish to download the artifact from the internet.

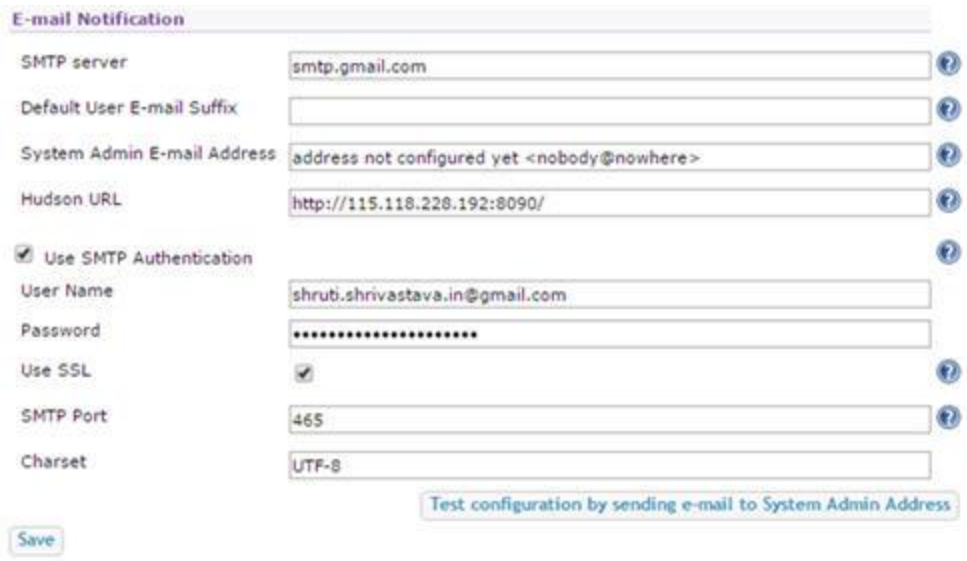
Configuring Email Notification

Email Notification section is shown in the end of the same webpage. User needs to configure the following fields:

Click on advanced button to see all the options related to Email Notification.

- **SMTP server:** SMTP Server stores the information about SMTP Server i.e. the IP number or fully qualified name of the server. For demonstration, In this tutorial, we will use Gmail’s SMTP server.
- **Default User E-mail Suffix**: An email suffix can be provided in this field which could be suffixed with the username and can be used to send the email notification.
- **System Admin E-mail Address**: Admin E-mail Address is used as a sender email id from which all the notifications would be sent.
- **Hudson URL**: If you are likely to publish reports or build information within the Email Notification, then Hudson URL needs to be provided. Hudson URL will be used to access the reports. A valid URL needs to be provided, however if all the receivers are connected to the intranet, then the IP address of the machine hosting Hudson can also be provided.
- **Use SMTP Authentication**: Enabling this option lets the username and password field appear for authentication purpose.
- **Use SS**L: User can activate SSL by selecting this option to connect with the SMTP Server.
- **SMTP Port:** User needs to provide the port number in this field which is used to communicate with the mail Server. If no port numbers are specified, then default port numbers are assigned.
- **Charset**: This field specifies the character set used to compose emails.

As we already mentioned that we would be using Gmail mail server to send email notification in this tutorial, refer the following screenshots and make the necessary changes in the Email Notification section.


E-mail Notification

SMTP server	smtp.gmail.com
Default User E-mail Suffix	
System Admin E-mail Address	address not configured yet <nobody@nowhere>
Hudson URL	http://115.118.228.192:8090/
✓ Use SMTP Authentication	
User Name	shruti.shrivastava.in@gmail.com
Password	••••••••••••••••••••••
Use SSL	✓
SMTP Port	465
Charset	UTF-8

Test configuration by sending e-mail to System Admin Address
Save

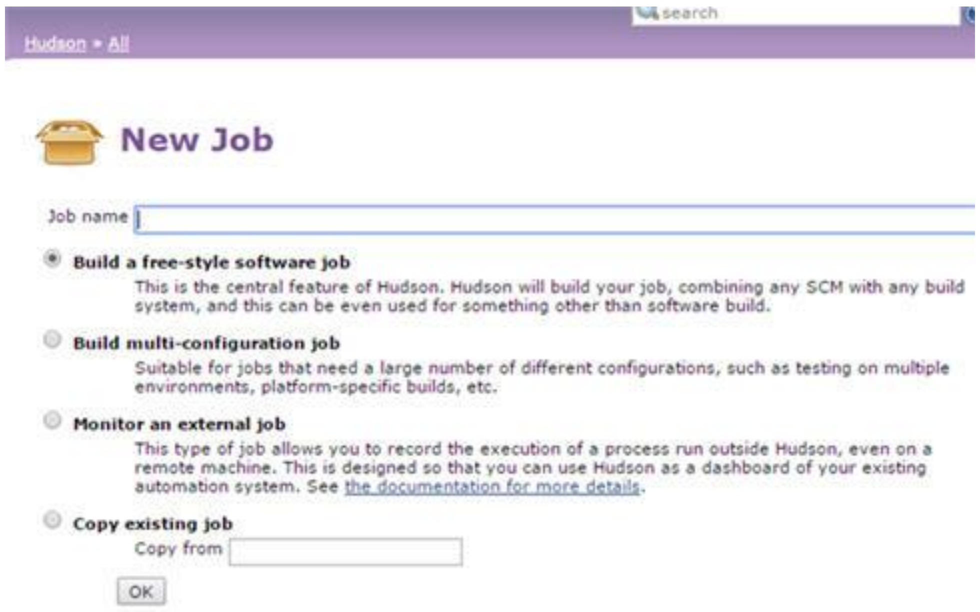
Click on the Save button in order to save all the newly made changes.

Creating Hudson Project

Now that we have installed and configured the Hudson on to our machines, we shall move ahead and create Hudson Projects. Like, Hudson configuration, we have several configuration options for a Hudson Project. In this tutorial, we would shed light on the most useful and popularly used options and extensions.

To create and configure a new Hudson Project, follow the steps ahead:

Click on the "New Job" option displayed in the left menu. The following page would open which displays the options related to project creation and project styles.



Job name

● **Build a free-style software job**
This is the central feature of Hudson. Hudson will build your job, combining any SCM with any build system, and this can be even used for something other than software build.

○ **Build multi-configuration job**
Suitable for jobs that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

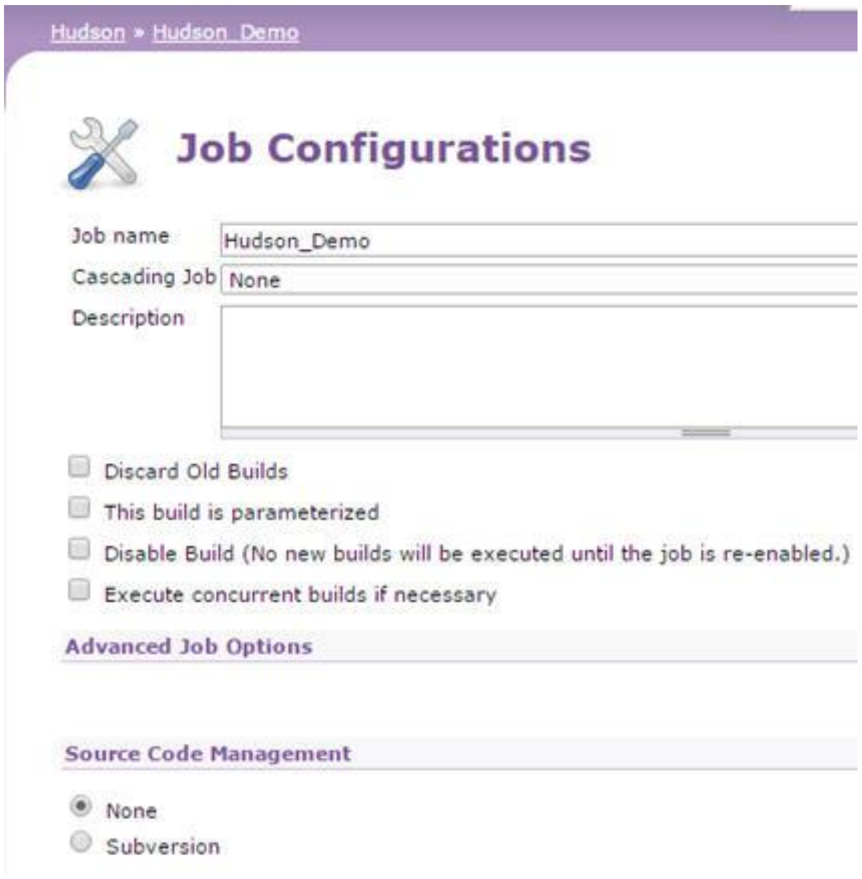
○ **Monitor an external job**
This type of job allows you to record the execution of a process run outside Hudson, even on a remote machine. This is designed so that you can use Hudson as a dashboard of your existing automation system. See the documentation for more details.

○ **Copy existing job**
Copy from

There are numerous styles in which the project/job can be created. Take a note that project and job can be used interchangeably as they both tend to mean the same thing.

- **Build a free-style software jo**b: This is the most commonly used method to create a new Hudson Job.
- **Build multi-configuration job**: This style of project is used to execute variety of jobs.
- **Monitor an external job**: This style of project monitors an external job.
- **Copy existing job**: In case if we have a project similar to an existing project, then this style can be helpful. All you have to do is to specify the existing job's name and the replica of this job would be created.

However, for this tutorial, we would create a free style Hudson project. Type in the name of the job you wish to create and click on the OK button. Clicking OK will land you to the Job's configuration page as shown below:



Job name: Hudson_Demo
Cascading Job: None
Description:

- [ ] Discard Old Builds
- [ ] This build is parameterized
- [ ] Disable Build (No new builds will be executed until the job is re-enabled.)
- [ ] Execute concurrent builds if necessary

[✓ None] Subversion

Configuring Hudson Project

Once, we have created the Hudson job, it's time to configure it. Like Hudson configuration, Hudson Job has also got various configuration settings. Let us discuss the important ones here.

To be specific, there are namely six types of settings to configure a job:

- **General Job Settings**: This section allows the user to mention the basic information about the job. User can submit the job description, disable the job, parameterize the job, trash the older builds and can execute more than one build for the same job concurrently.
- **Advanced Job Options**: This section allows the user to configure some advanced options.

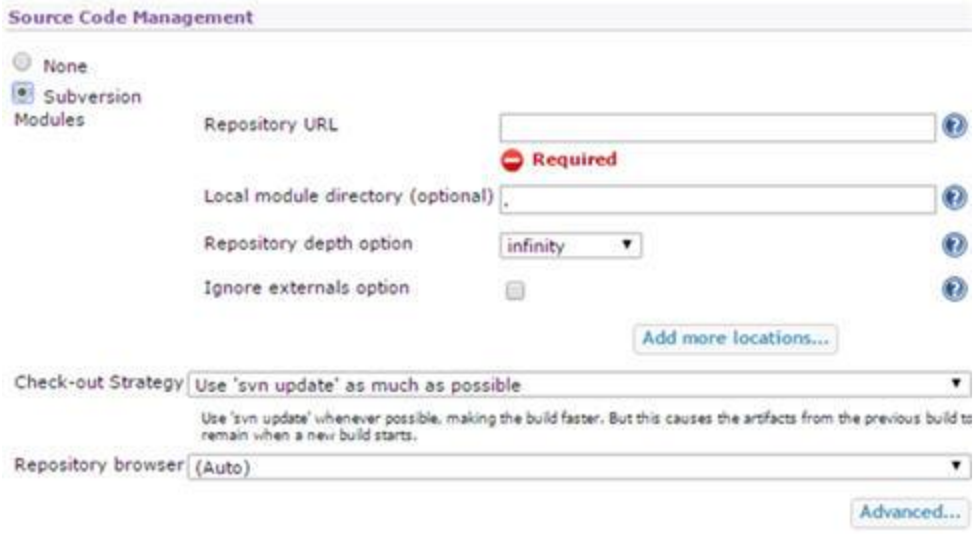
- **Source Code Management**: The section allows you to provide the settings related to Source Code Management system. Select "None" if no SCM is being used. Take a note that the user would be able to see only those SCM options whose plugin was installed at the time of Hudson installation. In order to add more SCM to the Hudson, user can visit the Manage Plugins page and can install the required plugins.
- **Build Triggers**: This section lets the user decide how to initiate the build execution.
- **Build**: This section lets the user provide the build mechanism settings.
- **Post-build Actions**: This section lets the user provide settings to the post build actions that would be executed once the build execution is completion.

Let us take a step ahead and configure the job with the necessary settings. User can leave the options under "General Job Settings" and "Advanced Job Options" to their default state.

Configuring Source Code Management

We have been talking much about creation of Hudson project in the above sections of this tutorial. Hudson project is customarily used with an actual project (Source Code) which is linked to a particular Source Code Management System. As mentioned in the beginning of this tutorial, Hudson has a great support to variety of SCMs. To name a few, Hudson supports CVS, Git, SVN etc. Thus, in this tutorial, we will configure Subversion (SVN) as SCM.

Step 1: Select "Subversion" option. As soon as the user selects Subversion, following options would appear.



Source Code Management

None
[✓ Subversion]

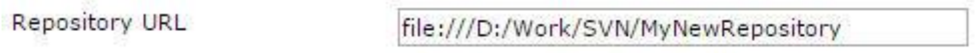
Modules

Repository URL:
Required
Local module directory (optional): .
Repository depth option: infinity
Ignore externals option: ☐
Add more locations...

Check-out Strategy: Use 'svn update' as much as possible
Use 'svn update' whenever possible, making the build faster. But this causes the artifacts from the previous build to remain when a new build starts.

Repository browser: (Auto)
Advanced...

Step 2: The next step is to provide the SVN's "Repository URL". As I have created a local repository, I would provide a local repository URL. A local repository can be created using Tortoise SVN.



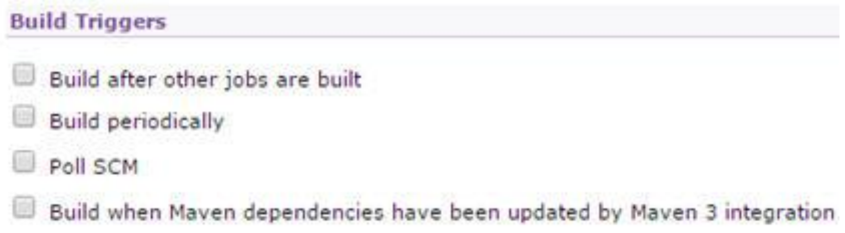
Repository URL: file:///D:/Work/SVN/MyNewRepository

Keep all the other settings in this section to default.

Selecting Build Triggers

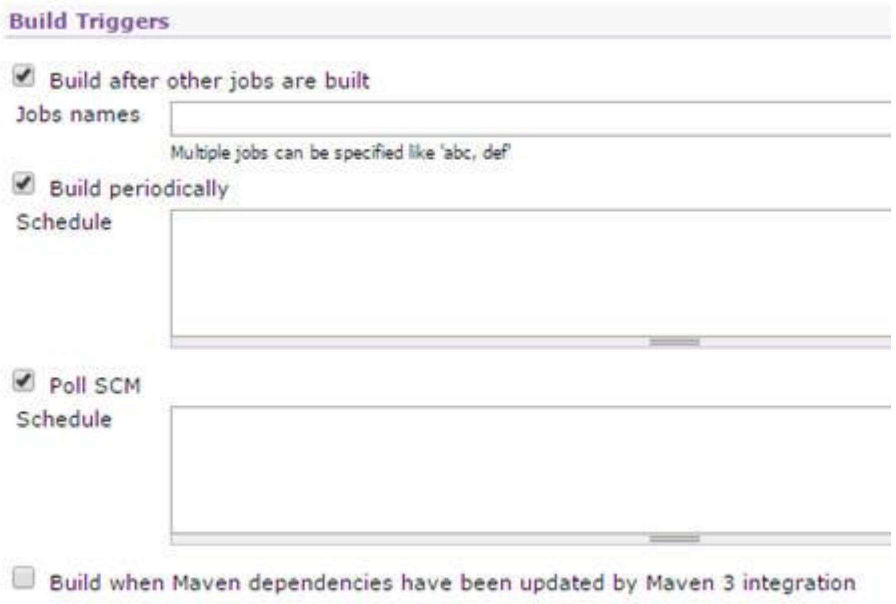
The next step is to configure the build triggers. Hudson allows you to set triggers to initiate the build execution process automatically. User can configure the job to build automatically if any other project/job is built. Alternatively, user can also set the build to execute periodically i.e. scheduling the build execution or user can also schedule a build to look for fresh commits in the SCM and trigger execution if any or the user can also set to initiate the build execute whenever there is an update in the maven dependencies provided your project is a Maven based project.

To set these options, all you have to do is select the desired build trigger. User is also leveraged to select more than one option at a time.


Build Triggers

- [ ] Build after other jobs are built
- [ ] Build periodically
- [ ] Poll SCM
- [ ] Build when Maven dependencies have been updated by Maven 3 integration

While selecting any of the above triggers, user might have to provide some additional information specific to the trigger type.


Build Triggers

- [✓] Build after other jobs are built

Jobs names

Multiple jobs can be specified like 'abc, def'

- [✓] Build periodically

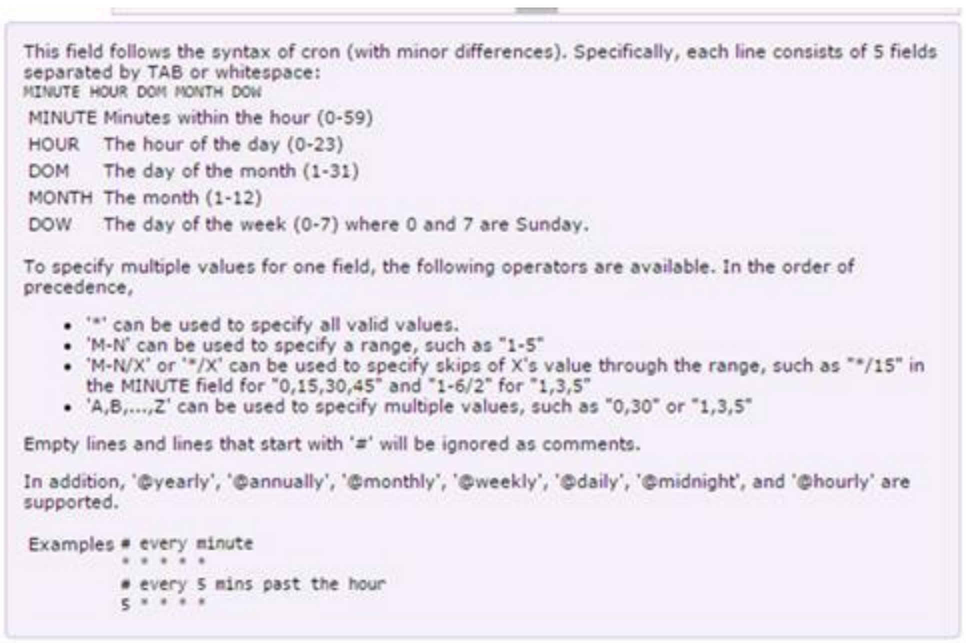
Schedule

- [✓] Poll SCM

Schedule

- [ ] Build when Maven dependencies have been updated by Maven 3 integration

- Build after other jobs are built: The name of the jobs which can trigger the execution of this job should be mentioned.
- Build periodically: The schedule should be mentioned. There is a specific protocol that needs to be followed to mention the schedule. More information on Schedule is shown below:



This field follows the syntax of cron (with minor differences). Specifically, each line consists of 5 fields separated by TAB or whitespace:
MINUTE HOUR DOM MONTH DOW

MINUTE	Minutes within the hour (0-59)
HOUR	The hour of the day (0-23)
DOM	The day of the month (1-31)
MONTH	The month (1-12)
DOW	The day of the week (0-7) where 0 and 7 are Sunday.

To specify multiple values for one field, the following operators are available. In the order of precedence,

- '*' can be used to specify all valid values.
- 'M-N' can be used to specify a range, such as "1-5"
- 'M-N/X' or '*/X' can be used to specify skips of X's value through the range, such as "*/15" in the MINUTE field for "0,15,30,45" and "1-6/2" for "1,3,5"
- 'A,B,...,Z' can be used to specify multiple values, such as "0,30" or "1,3,5"

Empty lines and lines that start with '#' will be ignored as comments.

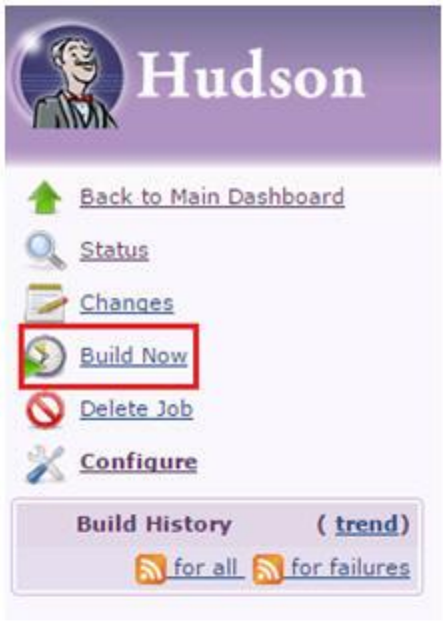
In addition, '@yearly', '@annually', '@monthly', '@weekly', '@daily', '@midnight', and '@hourly' are supported.

Examples
```
# every minute
* * * * *
# every 5 mins past the hour
5 * * * *
```

- Poll SCM: User needs to specify the schedule. The field acts same as that of “Build periodically”.
- Build when Maven dependencies have been updated by Maven 3 integration: This section doesn’t require any input to be submitted.

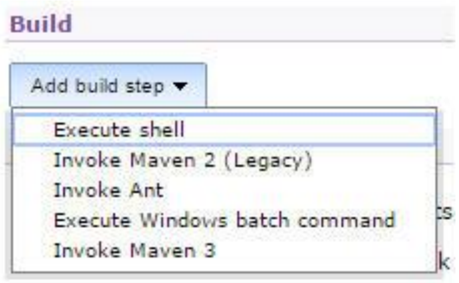
More information can be found by expanding the Help icons.

If the user doesn’t desire to set any of these build triggers, he/she can decide to build the job/project manually. All he/she has to do is to click on the “Build Now” link displayed in the left menu.



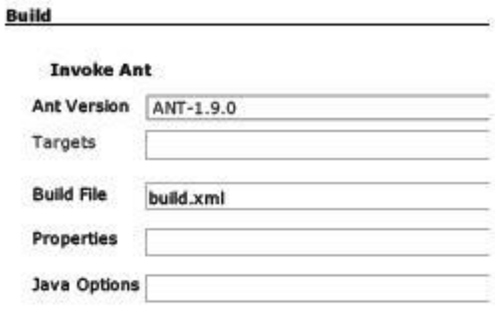
Invoking Build Steps

Now that we have seen all the basic steps to configure a build project, let us move ahead and add some more build steps. This section lets the user define his/her build with multiple build steps.



Each of the build steps has its own convention to define and invoke.

For instance, check out the ANT invocation below:



Build

Invoke Ant

Ant Version	ANT-1.9.0
Targets	
Build File	build.xml
Properties	
Java Options	

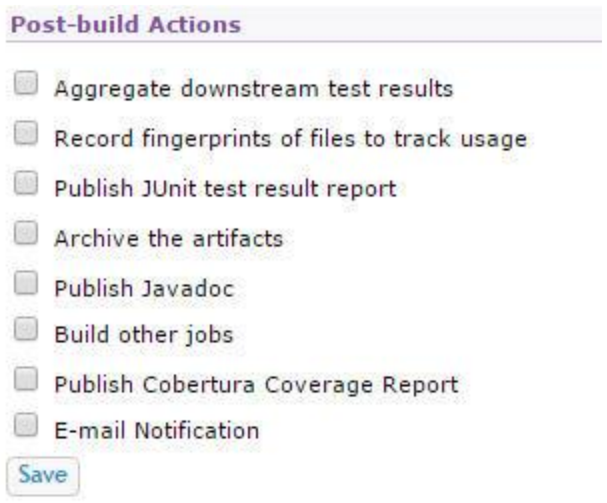
Configuring Post-build Actions

At times, it becomes necessary as well as vital to perform certain post build actions. Post build actions are nothing but some actions those are triggered once the build is executed. User is leveraged to trigger more than one post build action if he/she desires.

As we all know that the build execution statuses and reports are one of the most important artifacts or exit criteria for a Software development life cycle. Therefore, Hudson lets you publish the build execution report, generate documentations, generate executables/archives etc.

Test execution reports can be published and sent across to the stakeholders via Email. Results of this build can trigger the execution of another build.

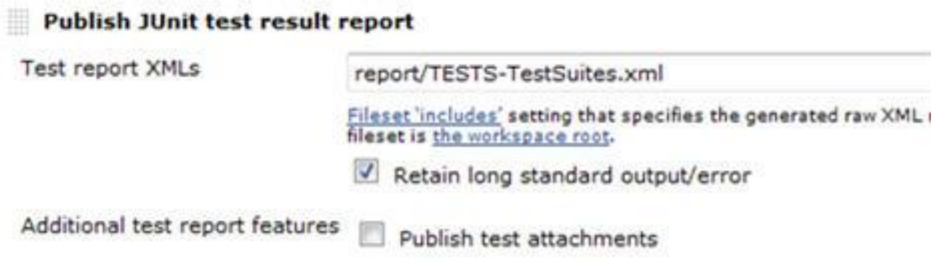
Post build Actions are many, let us take a moment to discuss the most basic ones.



Post-build Actions

☐ Aggregate downstream test results
☐ Record fingerprints of files to track usage
☐ Publish JUnit test result report
☐ Archive the artifacts
☐ Publish Javadoc
☐ Build other jobs
☐ Publish Cobertura Coverage Report
☐ E-mail Notification

#1. Aggregate downstream test results – The setting lets the user aggregate the test execution results of this job and downstream jobs together to produce more impactful test results. All user needs to do is to provide the name of the downstream job. In case if the user doesn't wish to provide any downstream job but still wishes to exploit the setting he can direct the Hudson to find all the downstream projects.
#2. Record fingerprints of files to track usage – The setting can be used by the user to track down where a particular file was used.
#3. Publish JUnit test result report – The setting allows the user to publish the JUnit test report by reading and understanding the custom report generated by JUnit. The JUnit test result report provides the user with a web interface to view the created reports. These reports can be sent over the mails to the stakeholders. To enable this option, all user is required to do is to provide the path to the custom report generated by JUnit.



Publish JUnit test result report

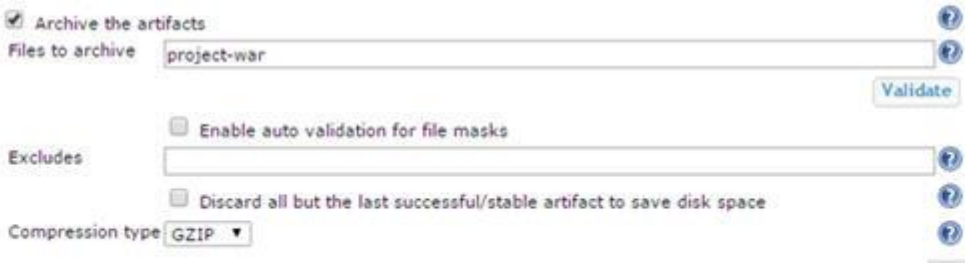
Test report XMLs: report/TESTS-TestSuites.xml

Fileset 'includes' setting that specifies the generated raw XML
fileset is the workspace root.

✓ Retain long standard output/error

Additional test report features: ☐ Publish test attachments

#4. Archive the artifacts – This setting lets the user create artifacts which can be distributed for further use. The artifact can be produced after every successful build. These artifacts can directly be accessed by the user over the web interface. Artifacts can be release executables in the form of war files, jar files, zipped or tar folders.



✓ Archive the artifacts

Files to archive: project-war

☐ Enable auto validation for file masks

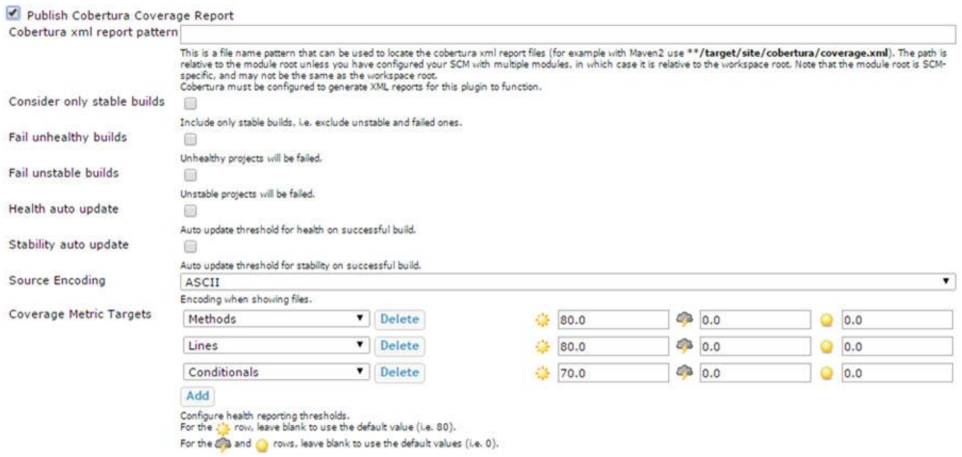
Excludes:

☐ Discard all but the last successful/stable artifact to save disk space

Compression type: GZIP

#5. Publish Javadoc – This setting lets you publish the java doc to customers and users on the Hudson web interface provided your project generates the java doc. To enable this option, user is required to provide the location of the Java Doc against Javadoc directory.
If user checks mark the option “Retain Javadoc for each successful build”, the newly generated Javadoc would be saved in the specified folder. Thus, all the Javadocs corresponding to the successful build would be maintained.

#6. Build other jobs – The setting lets the user trigger the execution of other jobs once this job is executed. User can trigger execution of more than one job at the same time. The setting can be helpful to execute unit test and integration test scenarios. User can even set the option to build other jobs even if this job fails (unstable).
#7. Publish Cobertura Coverage Report – Cobertura is a java based testing tool which analyzes the code coverage of your project i.e. it assess the percentage of code covered by the tests. Thus the setting allows the user to generate a report with Code coverage analysis. The setting requires a few parameters to be provided before you can get a fully fledged testing report on code coverage. Take a note that this setting doesn’t come by default i.e. it requires a plugin to be installed (Which we did at the time of installation as it is generally a part of the suggested plugins).



[✓] Publish Cobertura Coverage Report

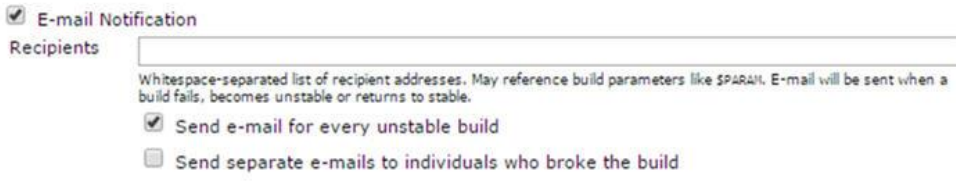
Cobertura xml report pattern	
	This is a file name pattern that can be used to locate the cobertura xml report files (for example with Maven2 use ****/target/site/cobertura/coverage.xml**). The path is relative to the module root unless you have configured your SCM with multiple modules, in which case it is relative to the workspace root. Note that the module root is SCM-specific, and may not be the same as the workspace root. Cobertura must be configured to generate XML reports for this plugin to function.
Consider only stable builds	☐ Include only stable builds, i.e. exclude unstable and failed ones.
Fail unhealthy builds	☐ Unhealthy projects will be failed.
Fail unstable builds	☐ Unstable projects will be failed.
Health auto update	☐ Auto update threshold for health on successful build.
Stability auto update	☐ Auto update threshold for stability on successful build.
Source Encoding	ASCII — Encoding when showing files.

Coverage Metric Targets

	☀	⛈	●
Methods	80.0	0.0	0.0
Lines	80.0	0.0	0.0
Conditionals	70.0	0.0	0.0

Configure health reporting thresholds.
For the ☀ row, leave blank to use the default value (i.e. 80).
For the ⛈ and ● rows, leave blank to use the default values (i.e. 0).

#8. E-mail Notification – Email Notification is one of the most important post build action. The option lets the user send the build notification email to the stakeholders (developers, testers, product owners etc.) by configuring their email ids. Hudson can send the email when the build is unstable, successful, failed etc. User can also set E-mail Notification triggers. The notification email can be send to more than one recipient at the same time just by providing a white-space between their email ids. Refer the below screenshot to check how these settings can be provided.
(Click on image to enlarge)



[✓] E-mail Notification

Recipients

Whitespace-separated list of recipient addresses. May reference build parameters like $PARAM. E-mail will be sent when a build fails, becomes unstable or returns to stable.

[✓] Send e-mail for every unstable build

☐ Send separate e-mails to individuals who broke the build

Notes:

1. User can anytime come back to this page and change the settings if required.
2. User can view the information about each option within the help icon associated with it.
3. User can add more post build actions with the help of plugins.

Conclusion:

In this tutorial, we made you acquainted with the concept of Continuous Integration. We also laid emphasis on its importance during a Software Development life cycle especially in a developer's or tester's life.

Next Tutorial #26: Moving ahead in the series, we would **discuss about some advanced Selenium concepts** that would directly or indirectly help in optimizing the Automation framework and brings more visibility to the users. Thus, in the next tutorial, we would discuss about the logging feature, its potential, debugging capabilities and much more.

Advanced Selenium

Most Popular Test Automation Frameworks with Pros and Cons of Each – Selenium Tutorial #20

In the last few Selenium tutorials, we discussed about various commonly and popularly used commands in WebDriver, handling web elements like Web Tables, Framesand handling exceptions in Selenium scripts. We discussed each of these commands with sample code snippets and examples so as to make you capable of using these commands effectively whenever you are encountered with similar situations. Amongst the commands we discussed in the previous tutorial, few of them owe utmost importance.

As we move ahead in the Selenium series, we would concentrate our focus towards **Automation Framework creation** in the next few upcoming tutorials. We would also shed light on various aspects of an Automation framework, types of Automation frameworks, benefits of using a framework and the basic components that constitutes an Automation framework.

What is Framework?
A framework is considered to be a combination of set protocols, rules, standards and guidelines that can be incorporated or followed as a whole so as to leverage the benefits of the scaffolding provided by the Framework.

Let us consider a real life scenario.
We very often use lifts or elevators. There are a few guidelines those are mentioned within the elevator to be followed and taken care off so as to leverage the maximum benefit and prolonged service from the system.

Thus, the users might have noticed the following guidelines:

- Keep a check on the maximum capacity of the elevator and do not get onto an elevator if the maximum capacity has reached.
- Press the alarm button in case of any emergency or trouble.
- Allow the passenger to get off the elevator if any before entering the elevator and stand clear off the doors.
- In case of fire in the building or if there is any haphazard situation, avoid the use of elevator.
- Do not play or jump inside the elevator.
- Do not smoke inside the elevator.
- Call for the help/assistance if door doesn't open or if the elevator doesn't work at all. Do not try to open the doors forcefully.

There can be many more rules or sets of guidelines. Thus, these guidelines if followed, makes the system more beneficial, accessible, scalable and less troubled for the users.

Now, as we are talking about "Test Automation Frameworks", let us move our focus towards them.

Test Automation Framework

A "Test Automation Framework" is scaffolding that is laid to provide an execution environment for the automation test scripts. The framework provides the user with various benefits that helps them to develop, execute and report the automation test scripts efficiently. It is more like a system that has created specifically to automate our tests.

In a very simple language, we can say that a framework is a constructive blend of various guidelines, coding standards, concepts, processes, practices, project hierarchies, modularity, reporting mechanism, test data injections etc. to pillar automation testing. Thus, user can follow these guidelines while automating application to take advantages of various productive results.

The advantages can be in different forms like ease of scripting, scalability, modularity, understandability, process definition, re-usability, cost, maintenance etc. Thus, to be able to grab these benefits, developers are advised to use one or more of the Test Automation Framework.

Moreover, the need of a single and standard Test Automation Framework arises when you have a bunch of developers working on the different modules of the same application and when we want to avoid situations where each of the developer implements his/her approach towards automation.

Note: Take a note that a testing framework is always application independent that is it can be used with any application irrespective of the complications (like Technology stack, architecture etc.) of application under test. **The framework should be scalable and maintainable.**

Advantage of Test Automation framework

1. Reusability of code
2. Maximum coverage
3. Recovery scenario
4. Low cost maintenance
5. Minimal manual intervention
6. Easy Reporting

Types of Test Automation Framework

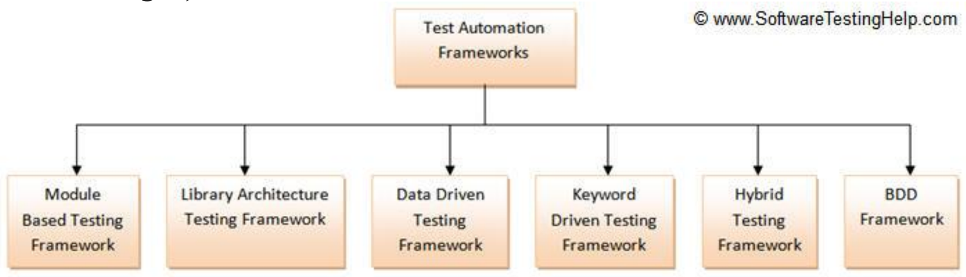
Now that we have a basic idea of what is an Automation Framework, in this section we would harbinger you with the various types of Test Automation Frameworks those are available in the market place. We would also try shed lights over their pros and cons and usability recommendations.

There is a divergent range of Automation Frameworks available now days. These frameworks may differ from each other based on their support to different key factors to do automation like reusability, ease of maintenance etc.

Let us discuss the few most popularly used Test Automation Frameworks:

1. Module Based Testing Framework
2. Library Architecture Testing Framework
3. Data Driven Testing Framework
4. Keyword Driven Testing Framework
5. Hybrid Testing Framework
6. Behavior Driven Development Framework

(click on image to view enlarged)


© www.SoftwareTestingHelp.com
Test Automation Frameworks
Module Based Testing Framework
Library Architecture Testing Framework
Data Driven Testing Framework
Keyword Driven Testing Framework
Hybrid Testing Framework
BDD Framework

Test Automation Frameworks

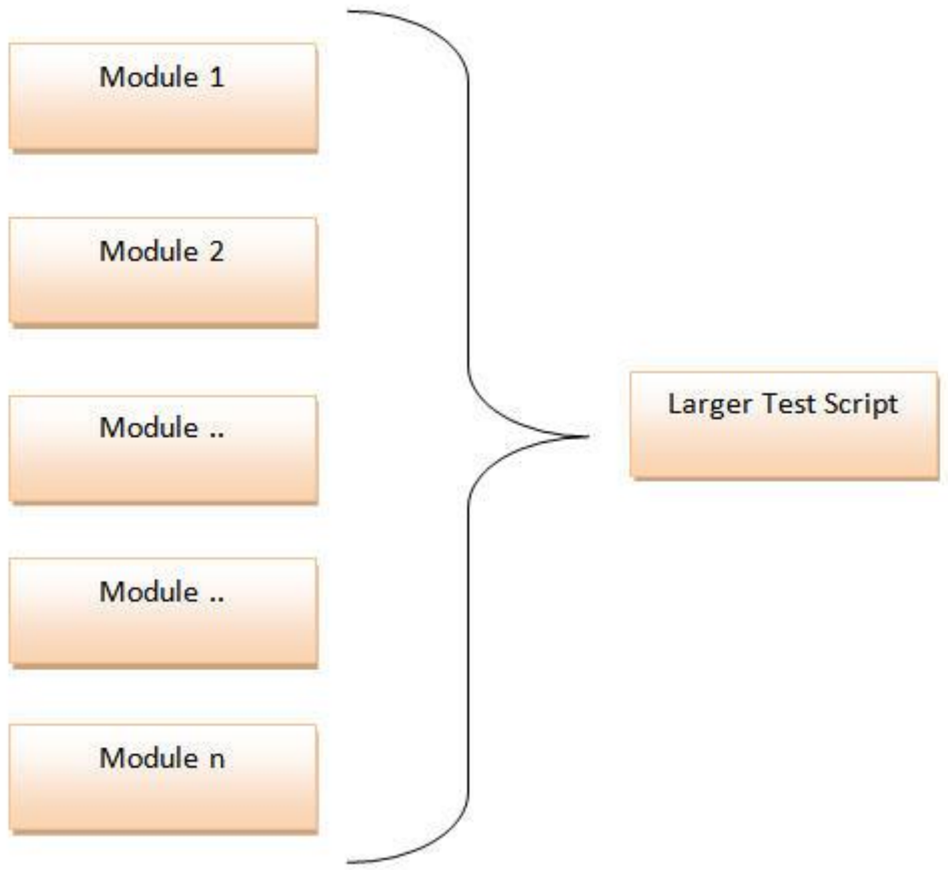
Let us discuss each of them in detail.

But before that I would also like to mention that despite having these framework, user is always leveraged to build and design his own framework which is best suitable to his/her project needs.

#1) Module Based Testing Framework

Module based Testing Framework is based on one of the popularly known OOPs concept – Abstraction. The framework divides the entire “Application Under Test” into number of logical and isolated modules. For each module, we create a separate and independent test script. Thus, when these test scripts taken together builds a larger test script representing more than one modules.

These modules are separated by an abstraction layer in such a way that the changes made in the sections of the application doesn’t yields affects on this module.



Pros:

1. The framework introduces high level of modularization which leads to easier and cost efficient maintenance.
2. The framework is pretty much scalable
3. If the changes are implemented in one part of the application, only the test script representing that part of the application needs to be fixed leaving all the other parts untouched.

Cons:

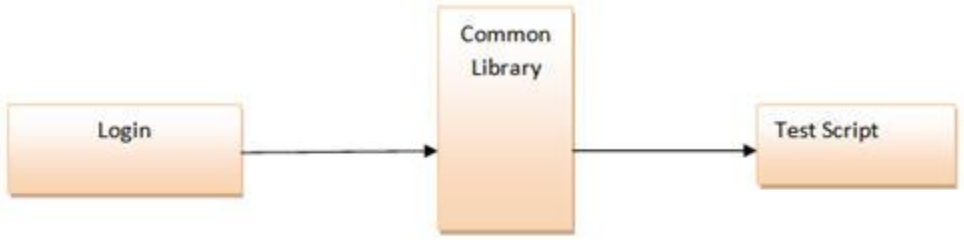
1. While implementing test scripts for each module separately, we embed the test data (Data with which we are supposed to perform testing) into the test scripts. Thus, whenever we are supposed to test with a different set of test data, it requires the manipulations to be made in the test scripts.

#2) Library Architecture Testing Framework

The Library Architecture Testing Framework is fundamentally and foundationally built on Module Based Testing Framework with some additional advantages. Instead of dividing the application under test into test scripts, we segregate the application into functions or rather common functions can be used by the other parts of the application as well. Thus we create a common library constituting of common functions for the application under test. Therefore, these libraries can be called within the test scripts whenever required.

The basic fundamental behind the framework is to determine the common steps and group them into functions under a library and call those functions in the test scripts whenever required.

<u>Example</u>: The login steps can be combined into a function and kept into a library. Thus all the test scripts those require to login the application can call that function instead of writing the code all over again.



Pros:

1. Like Module Based Framework, this framework also introduces high level of modularization which leads to easier and cost efficient maintenance and scalability too.
2. As we create common functions that can be efficiently used by the various test scripts across the Framework. Thus, the framework introduces a great degree of re-usability.

Cons:

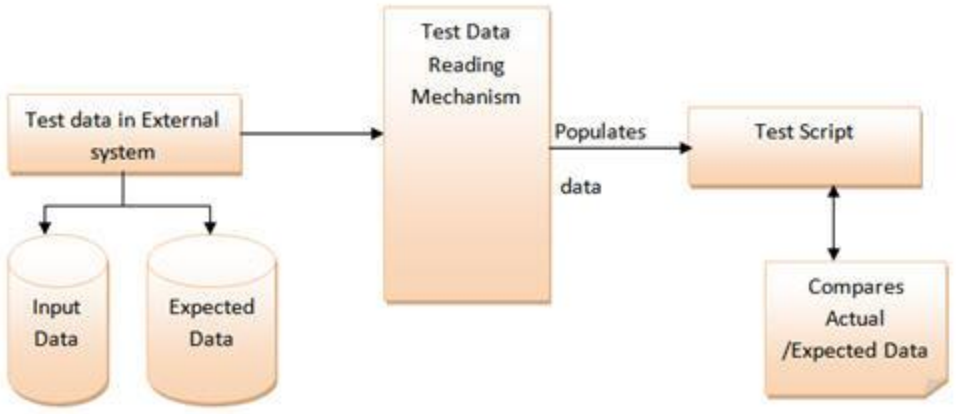
1. Like Module Based Framework, the test data is lodged into the test scripts, thus any change in the test data would require changes in the test script as well.
2. With the introduction of libraries, the framework becomes a little complicated.

#3) Data Driven Testing Framework

While automating or testing any application, at times it may be required to test the same functionality multiple times with the different set of input data. Thus, in such cases, we can't let the test data embedded in the test script. Hence it is advised to retain test data into some external data base outside the test scripts.

Data Driven Testing Framework helps the user segregate the test script logic and the test data from each other. It lets the user store the test data into an external database. The external databases can be property files, xml files, excel files, text files, CSV files, ODBC repositories etc. The data is conventionally stored in "Key-Value" pairs. Thus, the key can be used to access and populate the data within the test scripts.

Note: The test data stored in an external file can belong to the matrix of expected value as well as matrix of input values.

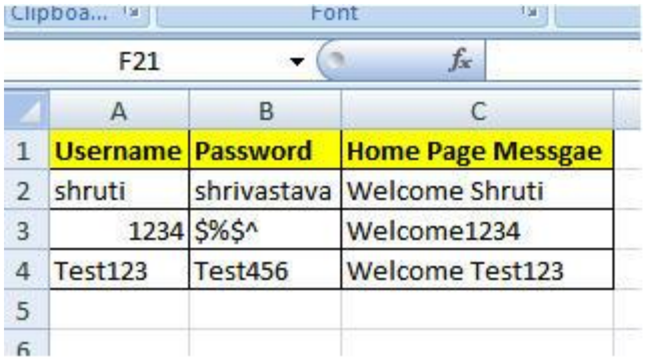

Test data in External system
Test Data Reading Mechanism
Populates data
Test Script
Input Data
Expected Data
Compares Actual /Expected Data

<u>Example</u>:

Let us understand the above mechanism with the help of an example.

Let us consider the "Gmail – Login" Functionality.

Step 1: First and the foremost step are to create an external file that stores the test data (Input data and Expected Data). Let us consider an excel sheet for instance.



	A	B	C
1	Username	Password	Home Page Messgae
2	shruti	shrivastava	Welcome Shruti
3	1234	$%$^	Welcome1234
4	Test123	Test456	Welcome Test123
5			
6			

Step 2: The next step is to populate the test data into Automation test Script. For this purpose several API's can be used to read the test data.

```
public void readTD(String TestData, String testcase) throws Exception {
            TestData=readConfigData(configFileName,"TestData",driver);
            testcase=readConfigData(configFileName,"testcase",driver);
                    FileInputStream td_filepath = new FileInputStream(TestData);
                    Workbook td_work =Workbook.getWorkbook(td_filepath);
                    Sheet td_sheet = td_work.getSheet(0);
                    if(counter==0)
                    {
          for (int i = 1,j = 1; i <= td_sheet.getRows()-1; i++){
                                        if(td_sheet.getCell(0,i).getContents().equalsIgnoreCase(testcase)){
              startrow = i;
                       arrayList.add(td_sheet.getCell(j,i).getContents());
                       testdata_value.add(td_sheet.getCell(j+1,i).getContents());}}
            for (int j = 0, k = startrow +1; k <= td_sheet.getRows()-1; k++){
                     if(td_sheet.getCell(j,k).getContents()==""){
                              arrayList.add(td_sheet.getCell(j+1,k).getContents());
                              testdata_value.add(td_sheet.getCell(j+2,k).getContents());}}
                     }
                     counter++;
}
```

The above method helps to read the test data and the below test step helps the user to type in the test data on the GUI.

element.sendKeys(obj_value.get(obj_index));

Pros:

1. The most important feature of this framework is that it considerably reduces the total number of scripts required to cover all the possible combinations of test scenarios. Thus lesser amount of code is required to test a complete set of scenarios.
2. Any change in the test data matrix would not hamper the test script code.
3. Increases flexibility and maintainability
4. A single test scenario can be executed altering the test data values.

Cons:

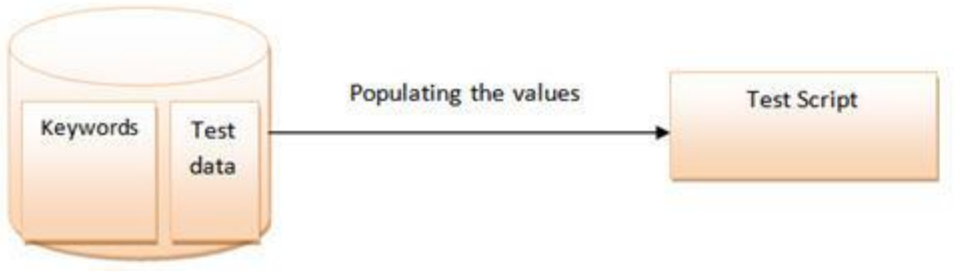
1. The process is complex and requires an extra effort to come up with the test data sources and reading mechanisms.
2. Requires proficiency in a programming language that is being used to develop test scripts.

#4) Keyword Driven Testing Framework

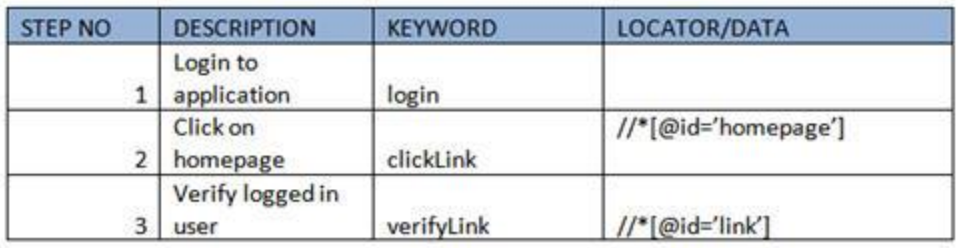
The Keyword driven testing framework is an extension to Data driven Testing Framework in a sense that it not only segregates the test data from the scripts, it also keeps the certain set of code belonging to the test script into an external data file.

These set of code are known as Keywords and hence the framework is so named. Key words are self-guiding as to what actions needs to be performed on the application.

The keywords and the test data are stored in a tabular like structure and thus it is also popularly regarded as Table driven Framework. Take a notice that keywords and test data are entities independent of the automation tool being used.



__Example__ **Test case of Keyword Driven Test Framework**



STEP NO	DESCRIPTION	KEYWORD	LOCATOR/DATA
1	Login to application	login	
2	Click on homepage	clickLink	//*[@id='homepage']
3	Verify logged in user	verifyLink	//*[@id='link']

In the above example keywords like login, clickLink and verifyLink are defined within the code. Depending upon the nature of application keywords can be derived. And all the keywords can be reused

multiple times in a single test case. Locator column contains the locator value that is used to identify the web elements on the screen or the test data that needs to be supplied.
All the required keywords are designed and placed in base code of the framework.

Pros:

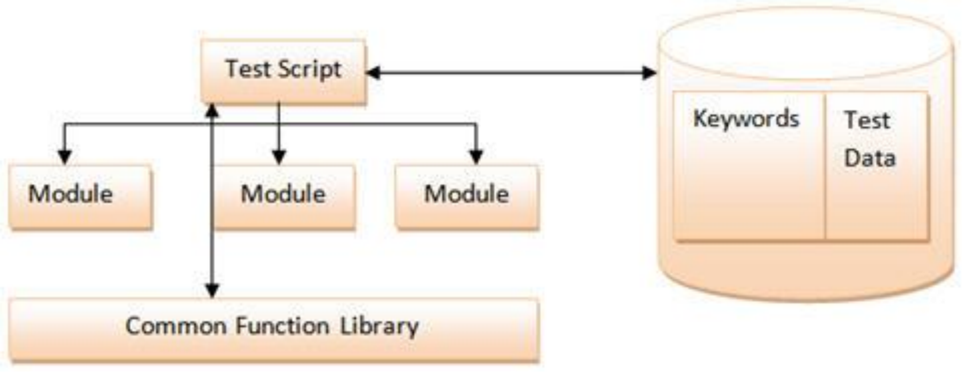
1. In addition to advantages provided by Data Driven testing, Keyword driven framework doesn't require the user to possess scripting knowledge unlike Data Driven Testing.
2. A single keyword can be used across multiple test scripts.

Cons:

1. The user should be well versed with the Keyword creation mechanism to be able to efficiently leverage the benefits provided by the framework.
2. The framework becomes complicated gradually as it grows and a number of new keywords are introduced.

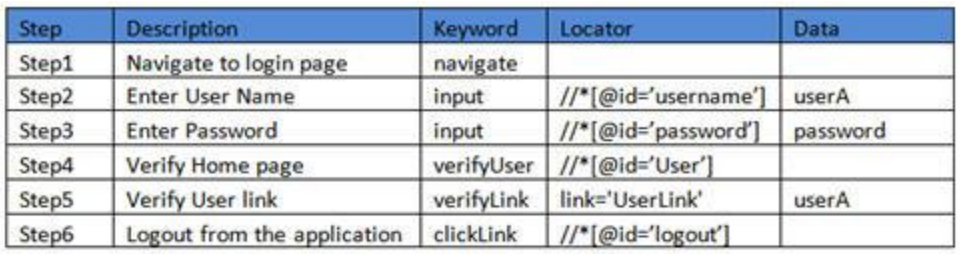
#5) Hybrid Testing Framework

As the name suggests, the Hybrid Testing Framework is a combination of more than one above mentioned frameworks. The best thing about such a setup is that it leverages the benefits of all kinds of associated frameworks.



<u>Example</u> of Hybrid Framework

Test sheet would contain both the keywords and the Data.



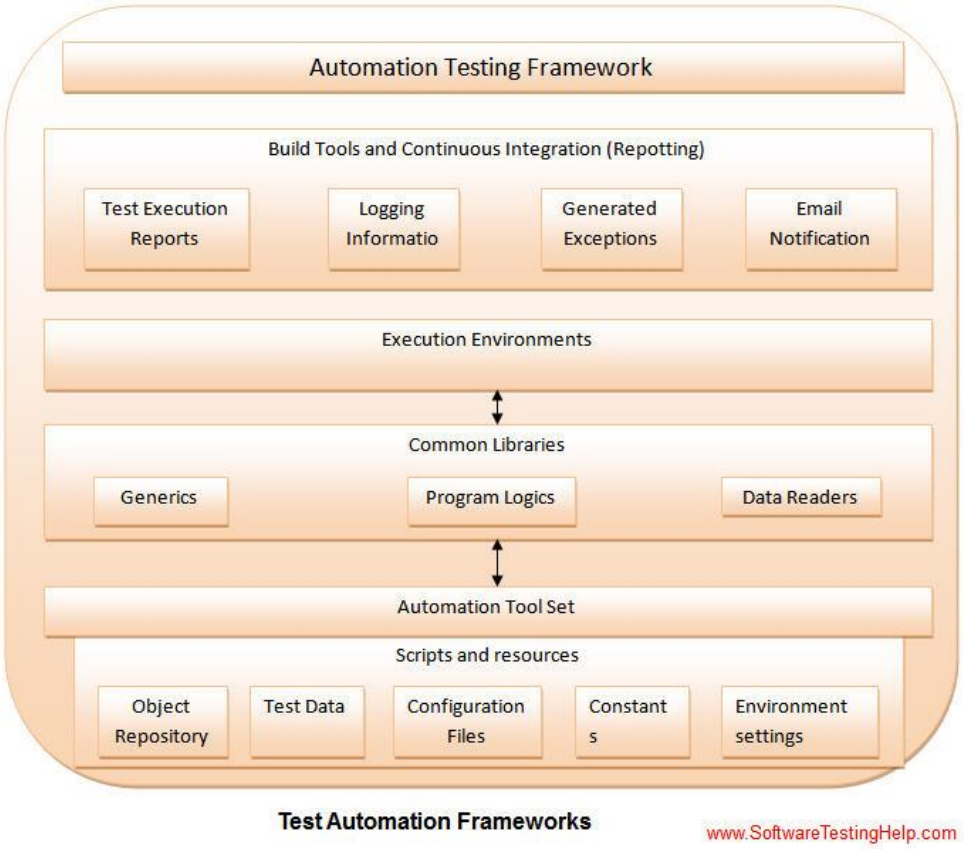
Step	Description	Keyword	Locator	Data
Step1	Navigate to login page	navigate		
Step2	Enter User Name	input	//*[@id='username']	userA
Step3	Enter Password	input	//*[@id='password']	password
Step4	Verify Home page	verifyUser	//*[@id='User']	
Step5	Verify User link	verifyLink	link='UserLink'	userA
Step6	Logout from the application	clickLink	//*[@id='logout']	

In the above example, keyword column contains all the required keywords used in the particular test case and data column drives all the data required in the test scenario. If any step does not need any input then it can be left empty.

#6) Behavior Driven Development Framework
Behavior Driven Development framework allows automation of functional validations in easily readable and understandable format to Business Analysts, Developers, Testers, etc. Such frameworks do not necessarily require the user to be acquainted with programming language. There are different tools available for BDD like cucumber, Jbehave etc. Details of BDD framework are discussed later in Cucumber tutorial. We have also discussed details on Gherkin language to write test cases in
Cucumber.

Components of Automation Testing Framework

(click on image to view enlarged)



Though the above pictorial representation of a framework is self-explanatory but we would still highlight a few points.

1. **Object Repository**: Object Repository acronym as OR is constituted of the set of locators types associated with web elements.
2. **Test Data:** The input data with which the scenario would be tested and it can be the expected values with which the actual results would be compared.
3. **Configuration File/Constants/ Environment Settings**: The file stores the information regarding the application URL, browser specific information etc. It is generally the information that remains static throughout the framework.

4. **Generics/ Program logics/ Readers**: These are the classes that store the functions which can be commonly used across the entire framework.
5. **Build tools and Continuous Integration**: These are the tools that aids to the frameworks capabilities to generate test reports, email notifications and logging information.

Conclusion

The frameworks illustrated above are the most popular frameworks used by the testing fraternity. There are various other frameworks also in the place. For all the further tutorials we would base on the **Data Driven Testing Framework**.

In this tutorial, we discussed about the basics of an Automation Framework. We also discussed about the types of frameworks available in the market.

Next Tutorial #27: In the next tutorial, we would briefly **introduce you with the sample framework, the MS Excel which would store the test data, excel manipulations etc.**

Till then feel free to ask your queries about automation frameworks.

Efficient Selenium Scripting and Troubleshoot Scenarios – Selenium Tutorial #27

In the previous tutorial, we discussed the technical implications **while implementing logging in a framework**. We discussed **log4j utility** at length. We discussed the basic components those constitute log4j from a usability perspective. With the Appenders and layouts, user is leveraged to choose the desired logging format/pattern and the data source/location.
In the current 27th tutorial in this comprehensive free selenium online training series, we would shift our focus towards a few trivial **yet important topics** that would guide us troubleshoot some recurrent problems. We may or may not use them in daily scripting but they would be helpful in the long run.
We would **discuss some advance concepts wherein we would deal with mouse and keyboard events, accessing multiple links by implementing lists**. So why not let's just start and briefly discuss these topics with the help of appropriate scenarios and code snippets.

JavaScript Executors

While automating a test scenario, there are certain actions those become an inherent part of test scripts.

These actions may be:

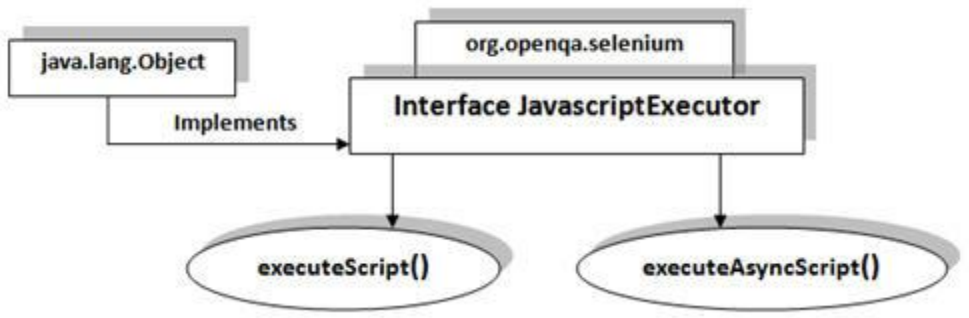
- Clicking a button, hyperlink etc.
- Typing in a text box
- Scrolling Vertically or Horizontally until the desired object is brought into view
- And many more

Now, it is evident from the earlier tutorials that the best way to automate such actions is by using Selenium commands.

But what if the selenium commands don't work?
Yes, it is absolutely possible that the very basic and elementary Selenium Commands don't work in certain situations.

That said, to be able to troubleshoot such situation, we shoulder JavaScript executors into the picture.

What are JavaScript Executors?



JavascriptExecutor interface is a part of org.openqa.selenium and implements java.lang.Object class. JavascriptExecutor presents the capabilities to execute JavaScript directly within the web-browser. To be able to execute the JavaScript, certain mechanisms in the form of methods along with a specific set of parameters are provided in its implementation.

Methods

executeScript (String script, args)

As the method name suggests, it executes the JavaScript within the current window, alert, frame etc (the window that the WebDriver instance is currently focusing on)

executeAsyncScript (String script, args)

As the method name suggests, it executes the JavaScript within the current window, alert, frame etc (the window that the WebDriver instance is currently focusing on)

The parameters and import statement are common to both the executor methods.

Parameters

Script – the script to be executed

Argument – the parameters that the script requires for its execution (if any)

Import statement

To be able to use JavascriptExecutors in our test scripts, we need to import the package using the following syntax:

import org.openqa.selenium.JavascriptExecutor;

Sample Code

#1) Clicking a web element

```
// Locating the web element using id
WebElement element = driver.findElement(By.id("id of the webelement"));

// Instantiating JavascriptExecutor
JavascriptExecutor js = (JavascriptExecutor)driver;

// Clicking the web element
```

```
js.executeScript("arguments[0].click();", element);
```

#2) Typing in a Text Box

```
// Instantiating JavascriptExecutor
JavascriptExecutor js = (JavascriptExecutor)driver;

// Typing the test data into Textbox
js.executeScript("document.getElementById('id of the element').value='test data';");
```

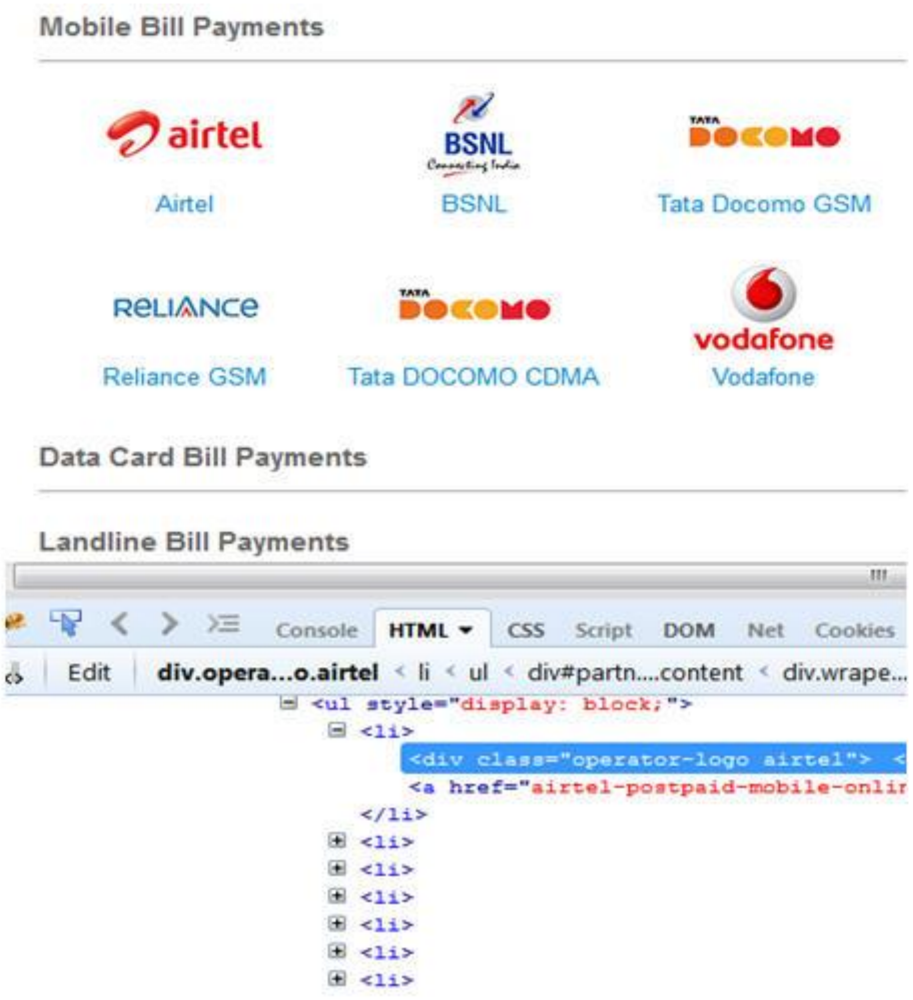
#3) Scrolling down until the web element is in the view

```
WebElement element=driver.findElement(By.xpath("//input[contains(@value,'Save')]"));

 // Instantiating the javascriptExecutor and scrolling into the view in the single test step
((JavascriptExecutor)driver).executeScript("arguments[0].scrollIntoView(true);",element);
```

You may find various other ways of writing down the code for accessing JavascriptExecutors.

Accessing multiple elements in a List

At times, we may come across elements of same type like multiple hyperlinks, images etc arranged in an ordered or unordered list. Thus, it makes absolute sense to deal with such elements by a single piece of code and this can be done using WebElement List. Refer the screenshot below to understand the elements I am talking about.



```
<ul style="display: block;">
  <li>
      <div class="operator-logo airtel"> <
      <a href="airtel-postpaid-mobile-onlir
    </li>
  <li>
  <li>
  <li>
  <li>
  <li>
  <li>
```

In the above image, we see that the various service providers belong to an unordered list. Thus, verification of click ability and visibility of these elements can be done by a single piece of code by using a list of elements.

Import statement

To be able to use WebElement list in our test scripts, we need to import the package using the following syntax:

import java.util.List;

Sample Code

```
// Storing the list
List <WebElement> serviceProviderLinks = driver.findElements(By.xpath("//div[@id='ServiceProvider']//ul//li"));

// Fetching the size of the list
int listSize = serviceProviderLinks.size();
for (int i=0; i<listSize; i++)
{

 // Clicking on each service provider link
serviceProviderLinks.get(i).click();

// Navigating back to the previous page that stores link to service providers
driver.navigate().back();
}
```

There are various requirements under which the lists can be used to verify the elements with suitable implementation changes.

Handling keyboard and mouse events

Handling Keyboard Events

As also said earlier, there are n numbers of ways to deal with the same problem statement in different contexts.

Thus, at times a necessity arises to deal with a problem by changing the conventional dealing strategy with a more advance strategy. I have witnessed cases where I could not deal with alerts and pop up etc. by selenium commands thus I had to opt for different java utilities to deal with it using keyboard strokes and mouse events.

Robot class is one such option to perform keyboard events and mouse events.

Let us understand the concept with the help of a scenario and its implementation.

Scenario:
Let us gather a situation where an unnecessary pop up appears on the screen which cannot be accepted or dismissed using Alert Interface, thus the only wise option we are left with is to close down the window using shortcut keys – "Alt + space bar + C". Let us see how we close the pop up using Robot Class.

Before, initiating the implementation, we should import the necessary package to be able to use Robot class within our test script.

Import Statement

import java.awt.Robot;

Sample Code

```
// Instantiating Robot class
Robot rb =new Robot();

// Calling KeyPress event
rb.keyPress(KeyEvent.VK_ALT);
rb.keyPress(KeyEvent.VK_SPACE);
rb.keyPress(KeyEvent.VK_C);

// Calling KeyRelease event
            rb.keyRelease(KeyEvent.VK_C);
rb.keyRelease(KeyEvent.VK_SPACE);
rb.keyRelease(KeyEvent.VK_ALT);
```

Robot class can also be used to handle mouse events but let us here look at the selenium's capabilities to handle mouse events.

Handling Mouse Events

WebDriver offers a wide range of interaction utilities that the user can exploit to automate mouse and keyboard events. Action Interface is one such utility which simulates the single user interactions.

Thus, we would witness Action Interface to mouse hover on a drop down which then opens a list of options in the next scenario.

Scenario:

1. Mouse Hover on the dropdown
2. Click on one of the items in the list options

Import Statement

import org.openqa.selenium.interactions.Actions;

Sample Code

```
// Instantiating Action Interface
Actions actions=<strong>new</strong> Actions(driver);

// howering on the dropdown
actions.moveToElement(driver.findElement(By.<em>id</em>("id of the dropdown"))).perform();

// Clicking on one of the items in the list options
WebElement subLinkOption=driver.findElement(By.id("id of the sub link"));
subLinkOption.click();
```

Conclusion

In this tutorial, we discussed some advance topics related to efficient scripting and to troubleshoot scenarios where the user is required to handle mouse and keyboard events. We also discussed how to store more than one web element in a list. I hope you would be able to troubleshoot these impediments if encountered.

Next Tutorial #28: For the upcoming tutorial in the series, we would discuss the **concept of Database testing using Selenium WebDriver**. We would witness the mechanism of database connection, hitting selenium queries and fetching the results through Selenium WebDriver Code.

Database Testing Using Selenium WebDriver and JDBC API – Selenium Tutorial #28

In our last Selenium tutorial we learned how to troubleshoot some recurrent problems in selenium scripts. We discussed some advance concepts wherein we would deal with mouse and keyboard events, accessing multiple links by implementing lists.

Moving ahead with our advance topics in Selenium training series, we would introduce you with the concept of **Database testing using Selenium WebDriver.**

We would discuss the basic processes like database connection, executing queries, fetching data and disconnecting database instances etc. We would also discuss various practical implications where we need Database testing with automation testing in order to test the **complete end-to-end scenarios.**

Before moving ahead with the technical implications associated with Automated Database testing. Let us discuss a few scenarios where we require performing Database testing along with the Automation Testing. But before that, I would like to affirm here that Database testing is a very peculiar type of testing whereas Selenium WebDriver is a tool used to simulate and automate user interactions with the Application UI.

So technically speaking we are not precisely performing Database Testing rather we are testing our application in conjunction with Database in order to ensure that the changes are reflected at both the ends thereby identifying defects early.

Absolutely all the web applications need a backend to store the Data. Databases like MySQL, Oracle, and SQL Server are reasonably popular these days.

Now motioning back to the original topic, let us discuss a few scenarios to exemplify the demand of Database testing along with Automation Testing.

Consider the following scenarios:

#1) At times, we are required to make sure that the data entered from the UI is consistently reflected at the database. Thus we retrieve the information from the Database and verify the retrieved information against the information supplied from the UI. For example, registration forms, user data, user profiles, updates and deletes of user data. Thus, the test scenario to automate can be “To verify that the user’s information is successfully saved into the database as soon as the user registers in the application”.

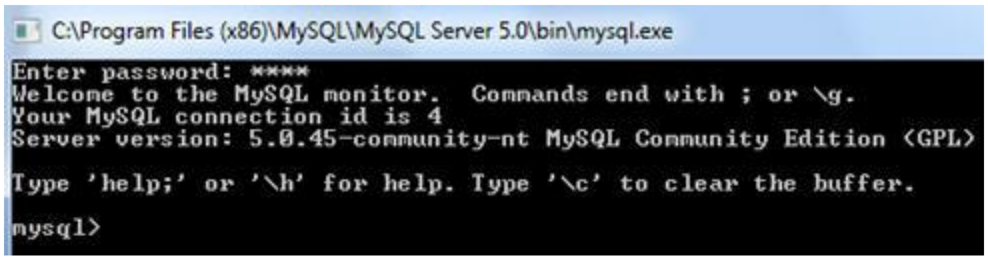
#2) Another use case of performing database testing with Selenium WebDriver may arise when the user is directed to load the test data or expected data from the Database. Thus, in such a case, user would make the

connection with the Database using a third party API, execute queries to retrieve data from the dataset and then asserting the data fetched from the Database with the actual data which is populated on the Application UI.
#3) Another user case is to perform associative Database Testing. Assume that we performed an operation on the application's UI, and we want to test the reflection in the Database. It may be a case that the impacted data resides in various tables of the database due to association. Therefore it is always advisable to test data reflection at all the impacted areas.
Selenium like I said simulates the user interactions with the application under test. It can simulate keyboard events, mouse actions etc. But if the user desires to automate anything outside the vicinity of browser – web application interactions, then selenium can't be of much help. Thus we require other tools or capabilities to perform end –to –end testing.

Thus, in all the above scenarios, we may require to perform Database Testing along with UI Automation. We may check business logics by manipulating the data and verifying its reflection. We may also check the technical aspects of the Database itself like soft delete, field validation etc.

Let us now move ahead with the actual implementation. Before developing Selenium WebDriver scripts to extract data from the data source, let us create a test data in the database. For this tutorial, we would use MySQL as a database.

Creation of test data in the Database
If you haven't downloaded the database yet, download it using the link – "http://dev.mysql.com/downloads/".
The user is expected to follow some basic steps to download and install the database.
Once the database is successfully installed, user can launch the MySQL Command Line Prompt which would look like the following screenshot. The application might ask the user to enter the password. The default password is "root".

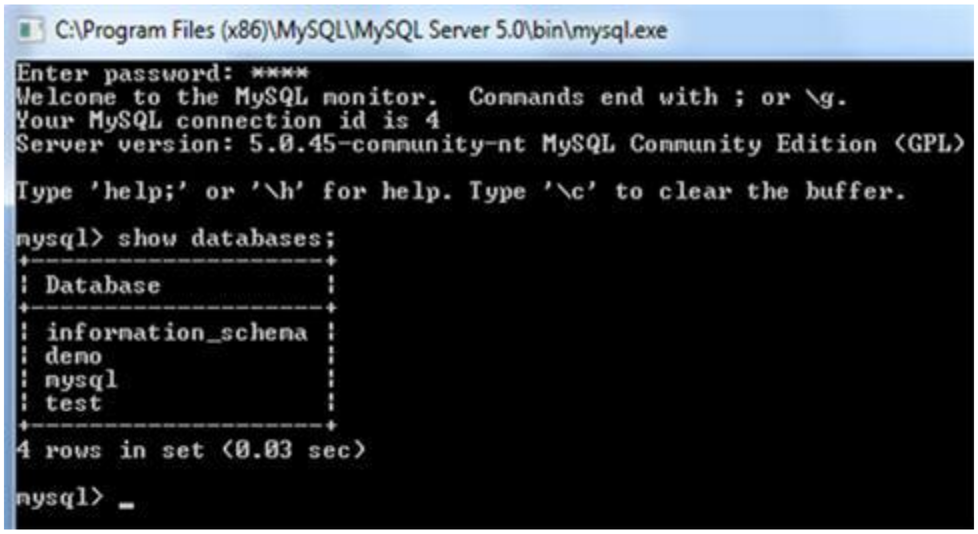


```
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.0.45-community-nt MySQL Community Edition (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Note: User can also find GUI based clients over the internet to connect with the database. To name a few, user can download and install Query Browser or Work Bench.
Creation of new Database
The next step is to create the test database with few tables and records stored in those tables in order to make connection with the database and execute queries.

Step 1) Type "show databases" to see all the already available databases

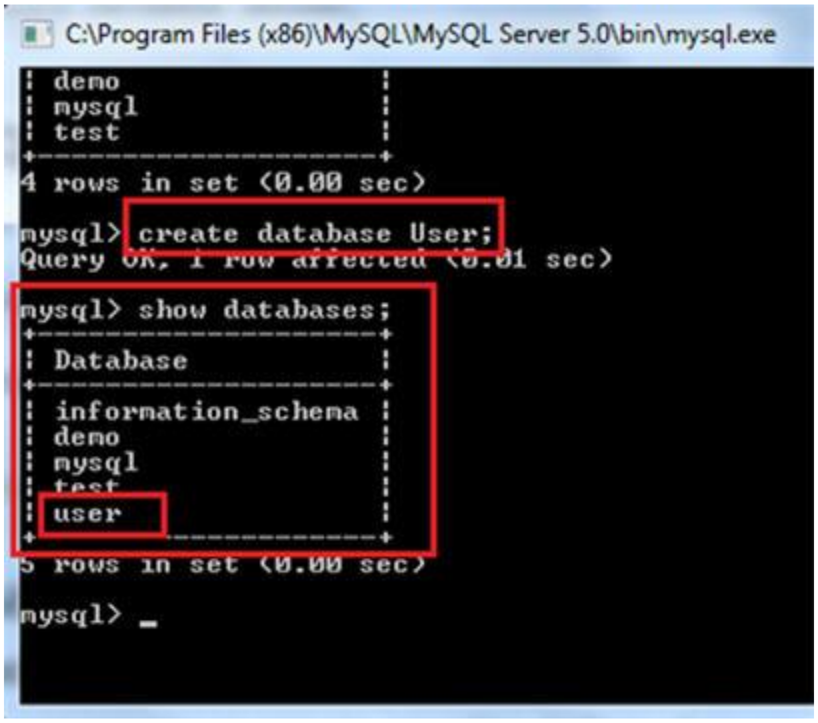
show databases;



```
C:\Program Files (x86)\MySQL\MySQL Server 5.0\bin\mysql.exe
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.0.45-community-nt MySQL Community Edition (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| demo               |
| mysql              |
| test               |
+--------------------+
4 rows in set (0.03 sec)

mysql> _
```

Step 2) Type "create database user;" to create a database named "user".

create database user;



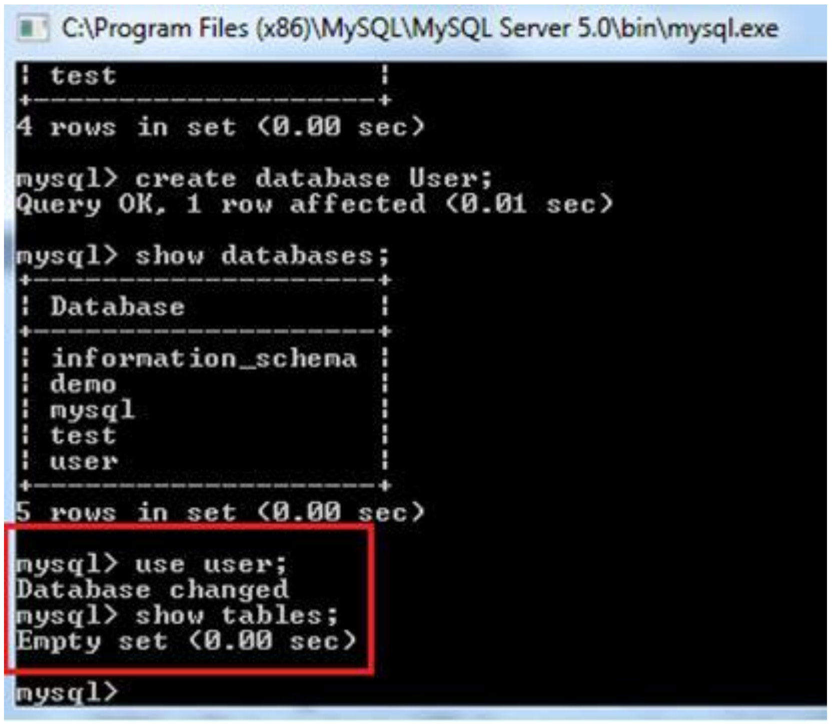
```
C:\Program Files (x86)\MySQL\MySQL Server 5.0\bin\mysql.exe
| demo               |
| mysql              |
| test               |
+--------------------+
4 rows in set (0.00 sec)

mysql> create database User;
Query OK, 1 row affected (0.01 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| demo               |
| mysql              |
| test               |
| user               |
+--------------------+
5 rows in set (0.00 sec)

mysql> _
```

Take a note that the database name as user is created and can be seen in the list of databases.

Step 3) Type "use user;" to select the newly created database. Also type "show tables;" to view all the tables available within the user database.

use user;

show tables;



```
| test                 |
+----------------------+
4 rows in set (0.00 sec)

mysql> create database User;
Query OK, 1 row affected (0.01 sec)

mysql> show databases;
+----------------------+
| Database             |
+----------------------+
| information_schema   |
| demo                 |
| mysql                |
| test                 |
| user                 |
+----------------------+
5 rows in set (0.00 sec)

mysql> use user;
Database changed
mysql> show tables;
Empty set (0.00 sec)

mysql>
```

Take a note that Empty set is shown in the result of the “show tables;” query as there were no tables available within the user database.

Let us now a few tables and add records in them.

Step 4) Type the following command to create a table with 4 fields/columns (userId, userName, userAge, userAddress).

create table userinfo

(

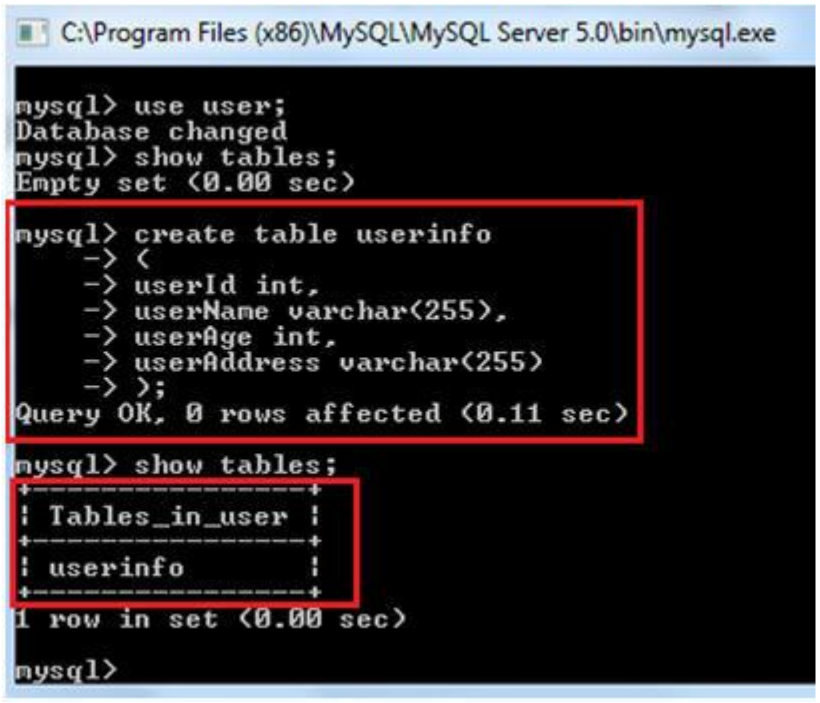
userId int,

userName varchar(255),

userAge int,

userAddress varchar(255)

);



```
C:\Program Files (x86)\MySQL\MySQL Server 5.0\bin\mysql.exe

mysql> use user;
Database changed
mysql> show tables;
Empty set (0.00 sec)

mysql> create table userinfo
    -> (
    -> userId int,
    -> userName varchar(255),
    -> userAge int,
    -> userAddress varchar(255)
    -> );
Query OK, 0 rows affected (0.11 sec)

mysql> show tables;
+----------------+
| Tables_in_user |
+----------------+
| userinfo       |
+----------------+
1 row in set (0.00 sec)

mysql>
```

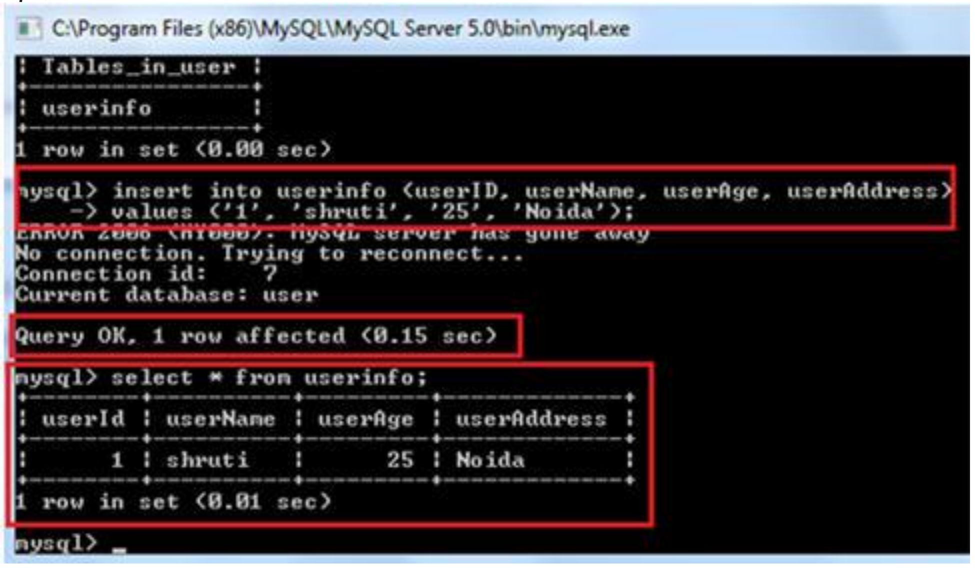
The next step is to add some data records in the “userinfo” table.

Step 5) Type the following command to insert data into the table a table for all the four fields 4 fields/columns (userId, userName, userAge, userAddress).

insert into userinfo (userID, userName, userAge, userAddress) values (‘1’, ‘shruti’, ’25’, ‘Noida’);

To view the added data, type the following command:

*select * from userinfo;*

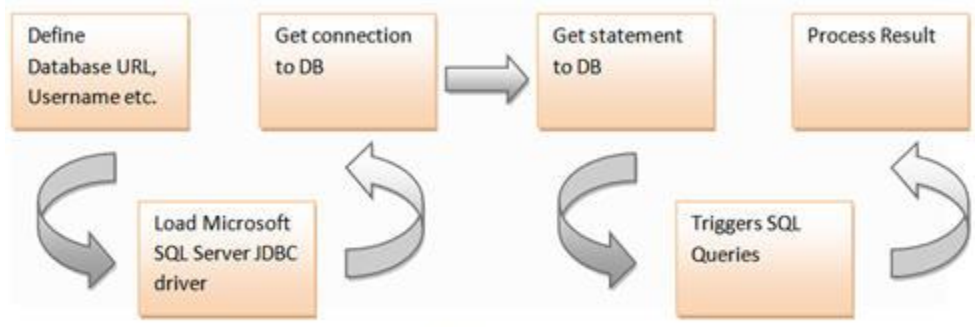


```
C:\Program Files (x86)\MySQL\MySQL Server 5.0\bin\mysql.exe
| Tables_in_user |
+----------------+
| userinfo       |
+----------------+
1 row in set (0.00 sec)

mysql> insert into userinfo (userID, userName, userAge, userAddress)
    -> values ('1', 'shruti', '25', 'Noida');
ERROR 2006 (HY000): MySQL server has gone away
No connection. Trying to reconnect...
Connection id:    7
Current database: user

Query OK, 1 row affected (0.15 sec)

mysql> select * from userinfo;
+--------+----------+---------+-------------+
| userId | userName | userAge | userAddress |
+--------+----------+---------+-------------+
|      1 | shruti   |      25 | Noida       |
+--------+----------+---------+-------------+
1 row in set (0.01 sec)

mysql> _
```

Similarly, you can add more data in your table and can create other tables as well.

Now, that we have created our database. We can move ahead and understand the **implementation of automated queries to fetch the records from the database.**
As we also iterated earlier, Selenium WebDriver is a tool for UI Automation. Thus, Selenium WebDriver alone is ineligible to perform database testing but this can be done using Java Database Connectivity API (JDBC). The API lets the user connect and interact with the data source and fetch the data with the help of automated queries. To be able to exploit the JDBC API, it is required to have Java Virtual Machine (JVM) running on the system.

JDBC Workflow



JDBC Workflow

We would keep our focus aligned with the following processes:

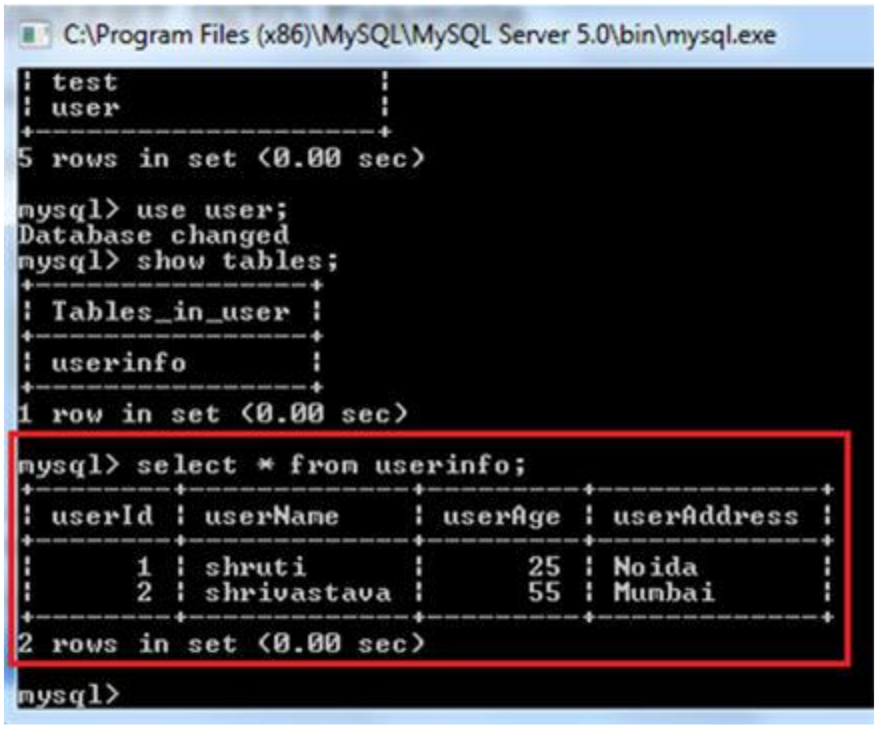
1. Creating connection with the database
2. Executing queries and update statements in order to extract/fetch data (CRUD Operations)
3. Using and manipulating the data extracted from the Database in the form of result set. (Result set is a collection of data organized in the rows and columns)
4. Disconnecting the database connection.

As said earlier, to be able to test database automatically from our Selenium WebDriver test scripts, we would connect with the Database via JDBC connectivity within our test scripts. Post to the connection, we can trigger as many CRUD (Create, Read, Update, and Delete) operations on the Database.

In this tutorial we would discuss "Read operation and its variants" and about their implementation in Selenium WebDriver script. But prior to that, let us check the test scenario manually using "MySQL command line".

Scenario

1) Open the Database server and connect to "user" database.
2) List down all the records from the "userinfo" table.
Syntax : *select * from userinfo;*



```
C:\Program Files (x86)\MySQL\MySQL Server 5.0\bin\mysql.exe
| test               |
| user               |
+--------------------+
5 rows in set (0.00 sec)

mysql> use user;
Database changed
mysql> show tables;
+----------------+
| Tables_in_user |
+----------------+
| userinfo       |
+----------------+
1 row in set (0.00 sec)

mysql> select * from userinfo;
+--------+--------------+---------+-------------+
| userId | userName     | userAge | userAddress |
+--------+--------------+---------+-------------+
|      1 | shruti       |      25 | Noida       |
|      2 | shrivastava  |      55 | Mumbai      |
+--------+--------------+---------+-------------+
2 rows in set (0.00 sec)

mysql>
```

3) Close the Database connection.

Notice that the read query will list down all the user data present in the userinfo table. The table is consisting of the following columns.

- userId
- username
- userAge
- userAddress

The result also shows that there is only a single data set present within the table.

Now, let us execute the same scenario using Java Class.

To be able to access Database, user is leveraged to choose amongst the diverse connector options available to connect with the Database. Most of the database connectors are freely distributed as “jar” files. As we are using MySQL as a data source, therefore we are required to download the jar file specific to MySQL.

The jar file can be downloaded from:

here or here.

Step 1: The first and the foremost step is to configure the project’s build path and add “mysql-connector-java-3.1.13-bin.jar” file as an external library.

Step 2: Create a java class named as “DatabaseTesingDemo”.

Step 3: Copy and paste the below code in the class created in the above step.

Code Sample

```
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class DatabaseTesingDemo {
        // Connection object
        static Connection con = null;
        // Statement object
        private static Statement stmt;
        // Constant for Database
URL
        public static String DB_URL = "jdbc:mysql://localhost:3306/user";
        // Constant for Database Username
        public static String DB_USER = "root";
        // Constant for Database Password
        public static String DB_PASSWORD = "root";

        @Before
        public void setUp() throws Exception {
                try{
                        // Make the database connection
                        String dbClass = "com.mysql.jdbc.Driver";
                        Class.forName(dbClass).newInstance();
                        // Get connection to DB
                        Connection con = DriverManager.getConnection(DB_URL, DB_USER, DB_PASSWORD);
                        // Statement object to send the SQL statement to the Database
                        stmt = con.createStatement();
                        }
                        catch (Exception e)
                        {
                            e.printStackTrace();
                        }
        }

        @Test
        public void test() {
                try{
                String query = "select * from userinfo";
                // Get the contents of userinfo table from DB
                ResultSet res = stmt.executeQuery(query);
```

```
            // Print the result untill all the records are printed
            // res.next() returns true if there is any next record else returns false
            while (res.next())
            {
                System.out.print(res.getString(1));
            System.out.print("\t" + res.getString(2));
            System.out.print("\t" + res.getString(3));
            System.out.println("\t" + res.getString(4));
            }
            }
            catch(Exception e)
            {
                e.printStackTrace();
            }
        }

        @After
        public void tearDown() throws Exception {
            // Close DB connection
            if (con != null) {
            con.close();
            }
        }
}
```

The output of the above code is:

```
1     shruti 25     Noida
2     shrivastava   55     Mumbai
```

Read Statement Variants

Where clause with single condition

*String query = "select * from userinfo where userId='" + 1 + "'";*

ResultSet res = stmt.executeQuery(query);

Output:

```
1     shruti 25     Noida
```

Where clause with multiple conditions

String Address ="Mumbai";

*String query = "select * from userinfo where userId='" + 2 + "' and userAddress='"+Address+"'";*

ResultSet res = stmt.executeQuery(query);

Output:

```
2     shrivastava   55     Mumbai
```

Display userId

String query = "select userId from userinfo";

ResultSet res = stmt.executeQuery(query);

Output:

1

2

Display userId with where clause

String Address ="Noida";

String query = "select userId,userName from userinfo where userAddress='"+Address+"'";

ResultSet res = stmt.executeQuery(query);

Output:

2

shrivastava

Thus, in the same way user can execute various queries on the database.

With this, Let us shed some light on result accessibility methods also.

Result Accessibility Methods:

Method name	**Description**
String getString()	Method is used to fetch the string type data from the result set
int getInt()	Method is used to fetch the integer type data from the result set
boolean getBoolean()	Method is used to fetch the boolean value from the result set
float getFloat()	Method is used to fetch the float type data from the result set
long getLong()	Method is used to fetch the long type data from the result set
short getShort()	Method is used to fetch the short type data from the result set
double getDouble()	Method is used to fetch the double type data from the result set
Date getDate()	Method is used to fetch the Date type object from the result set

Result Navigation Methods:

Method name	Description
boolean next()	Method is used to move to the next record in the result set
boolean previous()	Method is used to move to the previous record in the result set
boolean first()	Method is used to move to the first record in the result set
boolean last()	Method is used to move to the last record in the result set
boolean absolute(int rowNumber)	Method is used to move to the specific record in the result set

Conclusion

Through this tutorial, we tried to make you acquainted with the concept of **Automated Database Testing**. We clearly laid emphasis the technical implications and needs of Database Testing.

As our entire series was focused on Selenium, reader may get misled and can create an impression that this tutorial would teach to perform Database testing using Selenium, but like I mentioned number of times earlier, anything that lies outside the periphery of UI testing, cannot be handled by Selenium. Therefore we introduce Java Database Connectivity (JDBC) API in order to perform Database Testing by embedding the code within the Selenium WebDriver scripts.

JDBC makes it possible for the java class to connect with the Database, retrieve data from the database or for the matter of fact perform any of the CRUD operations, manipulate the resultant data and close the connection.

Thus, the tutorial constitutes of the basic sample implementation of the above mentioned process.

Next Tutorial #29: We will move ahead with advanced Selenium topics. In next tutorial we will cover Selenium GRID – which is used when you have to perform multi browser testing and you have large number of test cases.

How to Speed up Test Execution Using Selenium Grid – Selenium Tutorial #29

We are now close to the end of this comprehensive Selenium tutorials series. Next week, we will conclude this online Selenium Training series with "effort estimation of Selenium Projects" and "Selenium Interview questions and answers" tutorials.
Today, in this tutorial we will introduce you with **Selenium Grid – a *distributed test execution* environment to speed up the execution of a test pass.**

What is need of Selenium Grid?
As you go through entire Selenium WebDriver tutorials you will find out WebDriver will execute your test cases in a single machine.

Here are few problems with such a setup:

1. What if you want to execute your test cases for different Operating Systems?
2. How to run your test cases in different version of same browser?
3. How to run your test cases for multiple browsers?
4. Why a scenario should wait for execution of other test cases even if it does not depend upon any test cases?

All these problems are addressed in Selenium GRID.
As we proceed with the Selenium course, we will get the idea about how we can overcome to these problems. Basically Grid architecture is based on master slave architecture. Master machine distributes test cases to different slave machines.

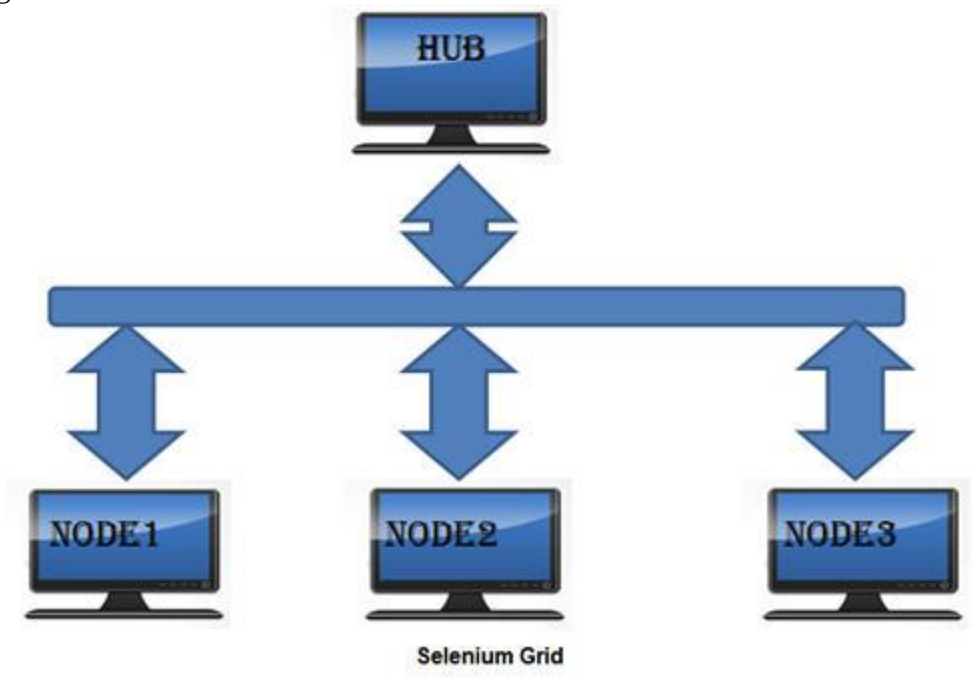
There are 2 versions of Grid available. Selenium Grid 2.0 is the latest from Selenium. Selenium 1.0 was the earlier version. Most of the Selenium experts prefer using Selenium Grid 2.0 as it is packed with new features. Selenium Grid 2.0 supports both Selenium RC and Selenium WebDriver scripts.

Benefits of Selenium Grid:

1. Selenium Grid gives the flexibility to distribute your test cases for execution.
2. Reduces batch processing time.
3. Can perform multi browser testing.
4. Can perform multi OS testing.

Basic terminology of Selenium Grid:

Hub: Hub is central point to the entire GRID Architecture which receives all requests. There is only one hub in the selenium grid. Hub distributes the test cases across each node.
Node: There can be multiple nodes in Grid. Tests will run in nodes. Each node communicates with the Hub and performs test assigned to it.



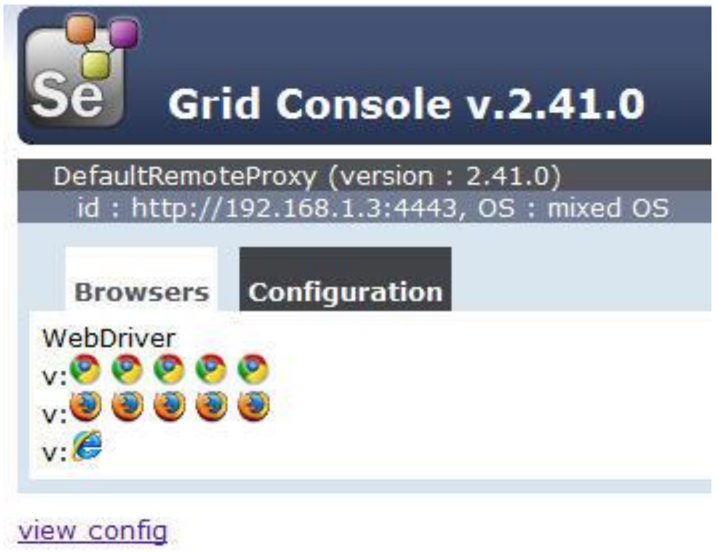
Install Selenium GRID

Step 1: Download Selenium Server jar file from Selenium's official website which is formerly known as Selenium RC Server and save it at any location on the local disk.
URL of selenium HQ: http://www.seleniumhq.org/download/
Step 2: Open command prompt and navigate to folder where the server is located. Run the server by using below command
java -jar selenium-server-standalone-2.41.0.jar -role hub
The hub will use the port 4444 by default. This port can be changed by passing the different port number in command prompt provided the port is open and has not been assigned a task.

Status can be checked by using the web interface: *http://localhost:4444/grid/console*
Step 3: Go to the other machine where you intend to setup Nodes. Open the command prompt and run the below line.

```
java -jar selenium-server-standalone-2.41.0.jar -role node -hub
http://localhost:4444/grid/register -port 5556
```

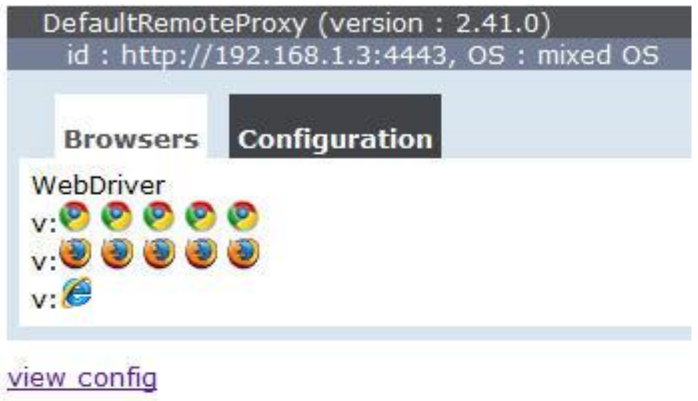
Run the selenium server in other machines to start nodes.


Grid Console v.2.41.0
DefaultRemoteProxy (version : 2.41.0)
id : http://192.168.1.3:4443, OS : mixed OS
Browsers
Configuration
WebDriver
v:
v:
v:
view config

Browser and Nodes:

After starting hub and nodes in each machine when you will navigate to GRID Console

You will find 5 Chrome, 5 Firefox and 1 IE browser under Browser section like below.

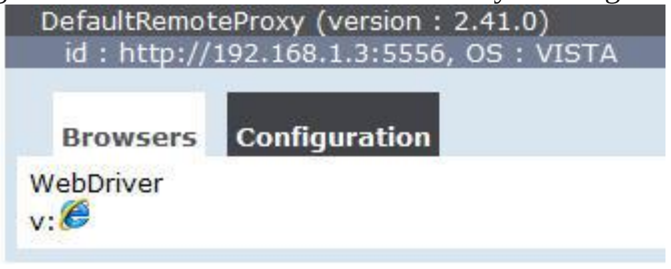

DefaultRemoteProxy (version : 2.41.0)
id : http://192.168.1.3:4443, OS : mixed OS
Browsers
Configuration
WebDriver
v:
v:
v:
view config

This indicates that by default you can use 5 Chrome, 5 Firefox and 1 IE browser.

For Instance if you want to use only IE you can start the node by using below command:

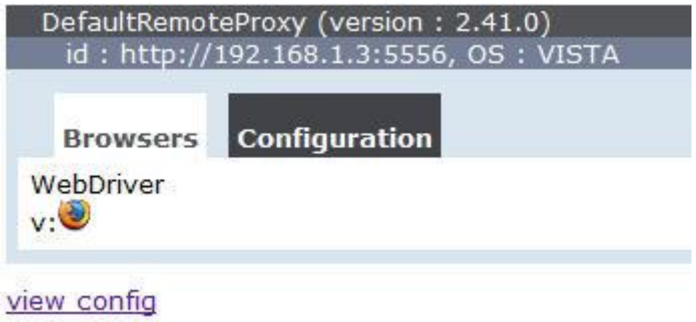
```
1 java -jar selenium-server-standalone-2.41.0.jar -role webdriver -hub
2 http://localhost:4444/grid/register -port 5556 -browser browserName=iexplore
```

Verify the browser Type along with other details in GRID Console by clicking on *view config*.


DefaultRemoteProxy (version : 2.41.0)
id : http://192.168.1.3:5556, OS : VISTA
Browsers
Configuration
WebDriver
v:

Similarly for Firefox:

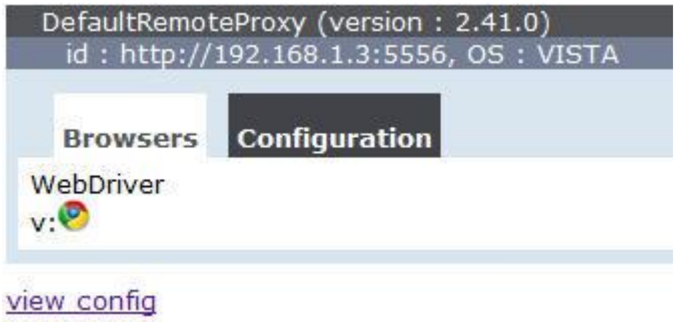
```
java -jar selenium-server-standalone-2.41.0.jar -role webdriver -hub
http://localhost:4444/grid/register -port 5556 -browser browserName=firefox
```



DefaultRemoteProxy (version : 2.41.0)
id : http://192.168.1.3:5556, OS : VISTA

For Chrome:

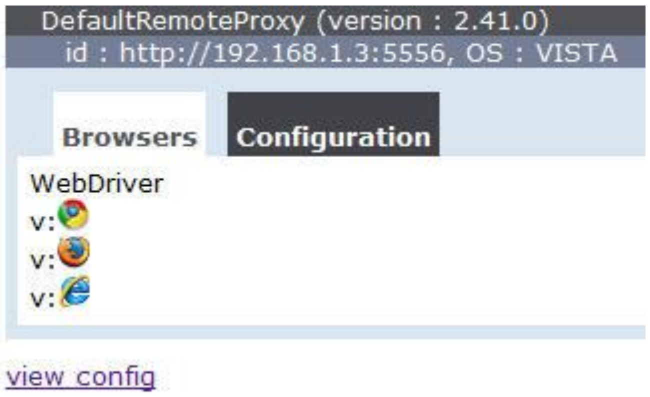
```
java -jar selenium-server-standalone-2.41.0.jar -role webdriver -hub
http://localhost:4444/grid/register -port 5556 -browser browserName=chrome
```



DefaultRemoteProxy (version : 2.41.0)
id : http://192.168.1.3:5556, OS : VISTA

There are few scenarios where you may need browser from each type i.e.: IE, Chrome and Firefox.

For instance you may need to use 1 IE and 1 Firefox and 1 Chrome browser



DefaultRemoteProxy (version : 2.41.0)
id : http://192.168.1.3:5556, OS : VISTA

```
java -jar selenium-server-standalone-2.41.0.jar -role webdriver -hub
http://localhost:4444/grid/register -port 5556 -browser browserName=iexplore
-browser browserName=firefox -browser browserName=chrome
```

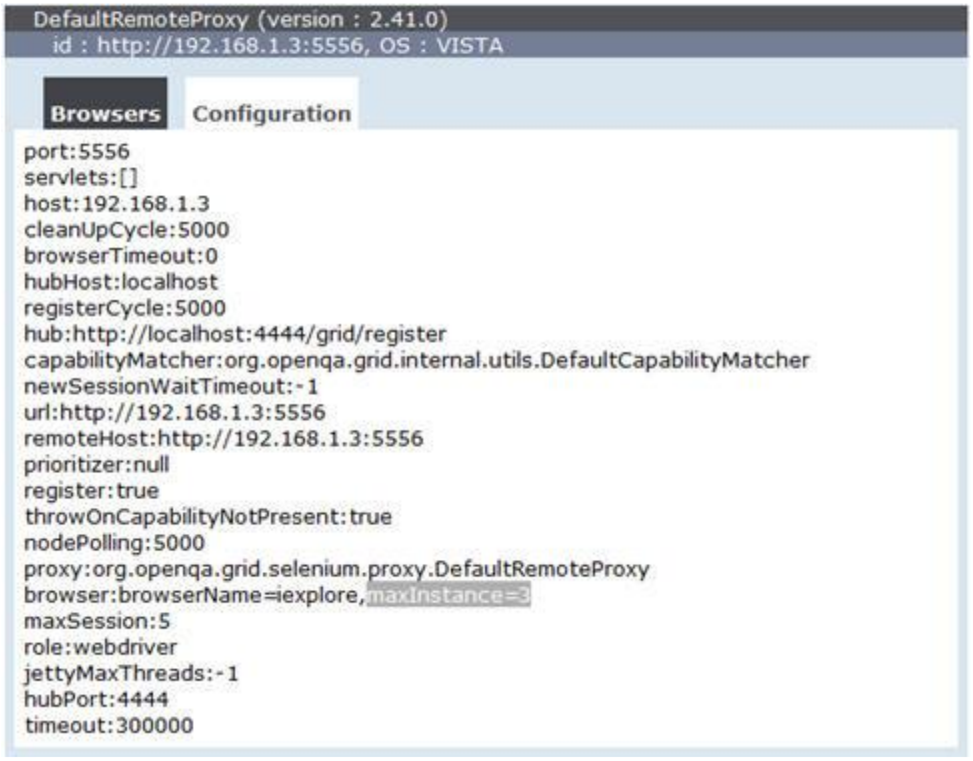
maxInstances:

maxInstance is used to limit the number of browser initialization in a node.

For example if you want to work with 2 Firefox and 2 IE then you can start the node using maxInstance.

```
java -jar selenium-server-standalone-2.41.0.jar -role webdriver -hub
http://localhost:4444/grid/register -port 5556 -browser browserName=firefox,maxInstance=3
```

Maximum instance can be verified under configuration tab.



DefaultRemoteProxy (version : 2.41.0)
id : http://192.168.1.3:5556, OS : VISTA
Browsers
Configuration

```
port:5556
servlets:[]
host:192.168.1.3
cleanUpCycle:5000
browserTimeout:0
hubHost:localhost
registerCycle:5000
hub:http://localhost:4444/grid/register
capabilityMatcher:org.openqa.grid.internal.utils.DefaultCapabilityMatcher
newSessionWaitTimeout:-1
url:http://192.168.1.3:5556
remoteHost:http://192.168.1.3:5556
prioritizer:null
register:true
throwOnCapabilityNotPresent:true
nodePolling:5000
proxy:org.openqa.grid.selenium.proxy.DefaultRemoteProxy
browser:browserName=iexplore,maxInstance=3
maxSession:5
role:webdriver
jettyMaxThreads:-1
hubPort:4444
timeout:300000
```

Similarly other browser instances can be configured using maxInstances.

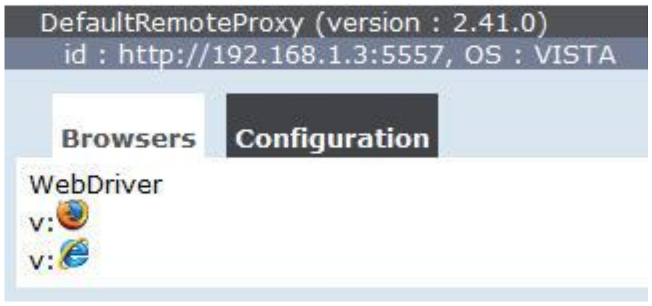
maxSession:

maxSession is used to configure how many number of browsers can be used parallel in the remote system.

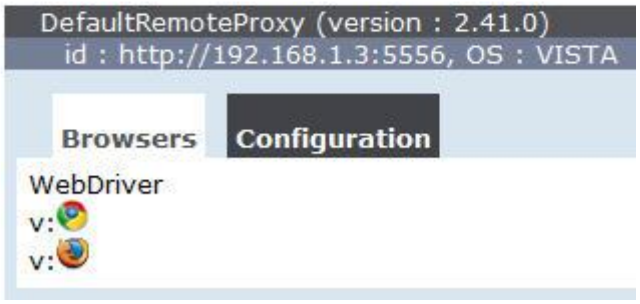
```
java -jar selenium-server-standalone-2.41.0.jar -role webdriver -hub
http://localhost:4444/grid/register -port 5556 -browser browserName=chrome,maxInstance=3
-browser browserName=firefox,maxInstance=3 –maxSession 3
```

Similarly you can start multiple nodes and configuration can be verified in the console.

NODE1:


DefaultRemoteProxy (version : 2.41.0)
id : http://192.168.1.3:5557, OS : VISTA

NODE2:


DefaultRemoteProxy (version : 2.41.0)
id : http://192.168.1.3:5556, OS : VISTA

Sample Grid Code:

Here I have used TestNG to run sample GRID test case.

Prerequisite: Create Hub and nodes as explained earlier and TestNG should be configured in eclipse.

Here I have taken sample test to login to Gmail and enter username and password

```
public class GridExample {
        @Test
        public void mailTest() throwsMalformedURLException{
                DesiredCapabilities dr=null;
                if(browserType.equals("firefox")){
                dr=DesiredCapabilities.firefox();
                dr.setBrowserName("firefox");
                dr.setPlatform(Platform.WINDOWS);
                }else{
                                    dr=DesiredCapabilities.internetExplorer();
                          dr.setBrowserName("iexplore");
                          dr.setPlatform(Platform.WINDOWS);
                  }
                  RemoteWebDriver driver=newRemoteWebDriver(new     URL("http://localhost:4444/wd/hub"), dr);
                  driver.navigate().to("http://gmail.com");
                  driver.findElement(By.xpath("//input[@id='Email']")) .sendKeys("username");
                  driver.findElement(By.xpath("//input[@id='Passwd']")) .sendKeys("password");
                  driver.close();
}
```

As in the example you have to use RemoteWebDriver if you are using GRID and you have to provide capabilities to the browser. You have to set the browser and platform as above.

In this example I have used platform as WINDOWS. You can use any platform as per your requirement.

Version of browser can also be set by using dr.setVersion("version")

For Instance you need to run this test serially in multiple browsers you have to configure your testng.xml .Below is the testng.xml suite for above test to run your test serially.

```
<?xml version="1.0" encoding="UTF-8"?>
<suite name="GRID SAMPLE TEST" thread-count="2">
  <test name="GRID TEST WITH SERIAL EXECTION WITH BROWSER IE">
  <parameter name ="browserType" value="IE"/>
    <classes>
      <class name ="GridExample"/>
    </classes>
  </test>

   <test name="GRID TEST WITH SERIAL EXECTION WITH BROWSER FF ">
   <parameter name ="browserType" value="firefox"/>
     <classes>
       <class name ="GridExample"/>
     </classes>
   </test>
</suite>
```

To run the test parallel, you have to change your testng.xml like below.

```
<?xml version="1.0" encoding="UTF-8"?>
<suite name="GRID SAMPLE TEST" parllel="tests" thread-count="3">
  <test name="GRID TEST WITH SERIAL EXECTION WITH BROWSER FF">
  <parameter name ="browserType" value="firefox"/>
    <classes>
      <class name ="GridExample"/>
    </classes>
  </test>
   <test name="GRID TEST WITH SERIAL EXECTION WITH BROWSER IE">
   <parameter name ="browserType" value="IE"/>
     <classes>
       <class name ="GridExample"/>
     </classes>
   </test>
</suite>
```

Here in the testng.xml you have to specify parameter as *parllel="tests" and thread-count="3"* describes the maximum number of threads to be executed in parallel.

Configuration using JSON file:

Grid can also be launched along with its configuration by using json configuration file.

Create a json file for having below configuration. Here I have created a json file named as grid_hub.json

```
{
 "host": null,
 "port": 4444,
 "newSessionWaitTimeout": -1,
 "servlets" : [],
 "prioritizer": null,
 "capabilityMatcher":   "org.openqa.grid.internal.utils.DefaultCapabilityMatcher",
  "throwOnCapabilityNotPresent": true,
   "nodePolling": 5000,

  "cleanUpCycle": 5000,
  "timeout": 300000,
  "maxSession": 5
}
```

Start the hub using below command

java -jar selenium-server-standalone-2.41.0.jar -role hub –hubConfig grid_hub.json

Similarly create different json file for different nodes as per required configuration.

Here is an example of json configuration file for node named as grid_node.json

```
{
 "capabilities":
     [
       {
         "browserName": "chrome",
         "maxInstances": 2
       },
       {
           "browserName": "firefox",
            "maxInstances": 2
         },
         {
            "browserName": "internet explorer",
            "maxInstances": 1
         }
       ],
    "configuration":
        {
        "nodeTimeout":120,
        "port":5555,

        "hubPort":4444,
        "hubHost":"localhost",

        "nodePolling":2000,

        "registerCycle":10000,
        "register":true,
        "cleanUpCycle":2000,
        "timeout":30000,
        "maxSession":5,
        }
}
```

<u>To start the node</u>

java -jar selenium-server-standalone-2.41.0.jar -role rc –nodeConfig grid_node.json

You can change all the configuration of browser, maxInstances, port, maxSession etc. in the json file.

You can provide browser version, platform in the json config file like below

```
{
  "browserName": "chrome","version":"8","platform":"Windows"
}
```

Conclusion:

It is advisable to **use Selenium Grid when you have to perform multi browser testing and you have large number of test cases.**

In this module we covered how to setup Grid hub and nodes along with how to run Grid test cases using testng.xml and json file.

<u>Next Tutorial #30</u>: **Automation testing with Selenium and Cucumber tool**. *Cucumber* is a BDD testing *tool* and Framework. We will learn features of Cucumber tool and its usage in real time scenario including **how to integrate Selenium WebDriver with Cucumber**.

Please post you queries related to Selenium Grid in comments below.

Automation Testing Using Cucumber Tool and Selenium – Selenium Tutorial #30

In last Selenium tutorial we introduced you to Selenium Gridwhich is **a *distributed test execution* environment to speed up the execution of a test pass**.
Now at the end of this comprehensive Selenium training series we are learning advanced Selenium testingand related concepts.
In this and the next tutorial we will be introducing you to the **Cucumber – a Behavior Driven Development (BDD) framework which is used with Selenium for performing acceptance testing.**

Cucumber Introduction:
Cucumber is a tool based on Behavior Driven Development (BDD) framework which is used to write acceptance tests for web application. It allows automation of functional validation in easily readable and understandable format (like plain English) to Business Analysts, Developers, Testers, etc.

Cucumber feature files can serve as a good document for all. There are many other tools like JBehave which also support BDD framework. Initially Cucumber was implemented in Ruby and then extended to Java framework. Both the tools support native JUnit.

Behavior Driven Development is extension of Test Driven Development and it is used to test the system rather than testing the particular piece of code. We will discuss more about the BDD and style of writing BDD tests.

Cucumber can be used along with Selenium, Watir, and Capybara etc. Cucumber supports many other languages like Perl, PHP, Python, .Net etc. In this tutorial we will concentrate on Cucumber with Java as a language.

Cucumber Basics:
In order to understand cucumber we need to know all the features of cucumber and its usage.

#1) Feature Files:
Feature files are essential part of cucumber which is used to write test automation steps or acceptance tests. This can be used as live document. The steps are the application specification. All the feature files ends with .feature extension.

Sample feature file:

Feature: Login Functionality Feature
In order to ensure Login Functionality works,
I want to run the cucumber test to verify it is working
Scenario: Login Functionality
Given user navigates to SOFTWARETETINGHELP.COM
When user logs in using Username as “USER” and Password “PASSWORD”
Then login should be successful
Scenario: Login Functionality
Given user navigates to SOFTWARETETINGHELP.COM
When user logs in using Username as “USER1” and Password “PASSWORD1”
Then error message should be thrown

#2) Feature:

This gives information about the high level business functionality (Refer to previous example) and the purpose of Application under test. Everybody should be able to understand the intent of feature file by reading the first Feature step. This part is basically kept brief.

#3) Scenario:

Basically a scenario represents a particular functionality which is under test. By seeing the scenario user should be able to understand the intent behind the scenario and what the test is all about. Each scenario should follow given, when and then format. This language is called as “gherkin”.

1. **Given:** As mentioned above, given specifies the pre-conditions. It is basically a known state.
2. **When**: This is used when some action is to be performed. As in above example we have seen when the user tries to log in using username and password, it becomes an action**.**
3. **Then:** The expected outcome or result should be placed here. For Instance: verify the login is successful, successful page navigation.
4. **Background:** Whenever any step is required to perform in each scenario then those steps needs to be placed in Background. For Instance: If user needs to clear database before each scenario then those steps can be put in background.
5. **And**: And is used to combine two or more same type of action.

Example:
Feature: Login Functionality Feature
Scenario: Login Functionality
Given user navigates to SOFTWARETETINGHELP.COM
When user logs in using Username as “USER”
And password as “password”
Then login should be successful
And Home page should be displayed

Example of Background:

Background:

Given user logged in as databases administrator

And all the junk values are cleared

#4) Scenario Outline:

Scenario outlines are used when same test has to be performed with different data set. Let’s take the same example. We have to test login functionality with multiple different set of username and password.

Feature: Login Functionality Feature

In order to ensure Login Functionality works,

I want to run the cucumber test to verify it is working

Scenario Outline: Login Functionality

Given user navigates to SOFTWARETESTINGHELP.COM

When user logs in using Username as <**username**> and Password <**password**>

Then login should be successful

Examples:

username	password
Tom	password1
Harry	password2
Jerry	password3

Note:

1. As shown in above example column names are passed as parameter to **When** statement.
2. In place of Scenario, you have to use Scenario Outline.
3. Examples are used to pass different arguments in tabular format. Vertical pipes are used to separate two different columns. Example can contain many different columns.

#5) Tags:

Cucumber by default runs all scenarios in all the feature files. In real time projects there could be hundreds of feature file which are not required to run at all times.

For instance: Feature files related to smoke test need not run all the time. So if you mention a tag as smokeTest in each feature file which is related to smoke test and run cucumber test with @SmokeTest tag . Cucumber will run only those feature files specific to given tags. Please follow the below example. You can specify multiple tags in one feature file.

Example of use of single tags:

@SmokeTest

Feature: Login Functionality Feature

In order to ensure Login Functionality works,
I want to run the cucumber test to verify it is working
Scenario Outline: Login Functionality
Given user navigates to SOFTWARETESTINGHELP.COM
When user logs in using Username as <**username**> and Password <**password**>
Then login should be successful
Examples:
username	password
Tom	password1
Harry	password2
Jerry	password3
Example of use of multiple tags:
As shown in below example same feature file can be used for smoke test scenarios as well as for login test scenario. When you intend to run your script for smoke test then use @SmokeTest. Similarly when you want your script to run for Login test use @LoginTest tag.

Any number of tags can be mentioned for a feature file as well as for scenario.

@SmokeTest @LoginTest
Feature: Login Functionality Feature
In order to ensure Login Functionality works,
I want to run the cucumber test to verify it is working
Scenario Outline: Login Functionality
Given user navigates to SOFTWARETETINGHELP.COM
When user logs in using Username as <username> and Password <password>
Then login should be successful
Examples:
username	password
Tom	password1
Harry	password2
Jerry	password3
Similarly you can specify tags to run specific scenario in a feature file. Please check below example to run specific scenario.

Feature: Login Functionality Feature
In order to ensure Login Functionality works,
I want to run the cucumber test to verify it is working

@positiveScenario
Scenario: Login Functionality
Given user navigates to SOFTWARETETINGHELP.COM
When user logs in using Username as “USER” and Password “PASSWORD”
Then login should be successful
@negaviveScenario
Scenario: Login Functionality
Given user navigates to SOFTWARETETINGHELP.COM
When user logs in using Username as “USER1” and Password “PASSWORD1”
Then error message should throw

#6) Junit Runner:

To run the specific feature file cucumber uses standard Junit Runner and specify tags in @Cucumber. Options. Multiple tags can be given by using comma separate. Here you can specify the path of the report and type of report you want to generate.

Example of Junit Runner:

```
import cucumber.api.junit.Cucumber;</pre>
import org.junit.runner.RunWith;
 @RunWith(Cucumber.class)
@Cucumber.Options(format={"SimpleHtmlReport:report/smokeTest.html"},tags={"@smokeTest"})
Public class JUnitRunner {
}
```

Similarly you can give instruction to cucumber to run multiple tags. Below example illustrates how to use multiple tags in cucumber to run different scenarios.

```
import cucumber.api.junit.Cucumber;
 import org.junit.runner.RunWith;
  @RunWith(Cucumber.class)
 @Cucumber.Options(format={"SimpleHtmlReport:report/smokeTest.html"},tags={"@smokeTest",”@LoginTest”})
 Public class JUnitRunner {
}
```

#7) Cucumber Report:

Cucumber generates its own html format. However better reporting can be done using Jenkins or bamboo tool. Details of reporting are covered in next topic of cucumber.

Cucumber Project Setup:

Detail explanation of cucumber project set up is available separately in next tutorial. Please refer to Cucumber Tutorial Part2 from more information about project setup. Remember there is no extra software installations required for cucumber.

Implementation of Feature file:
We have to implement these steps in Java in order to test the feature files. Need to create a class which contains those given, when and then statements. Cucumber uses its annotations and all the steps are embedded in those annotations (given, when, then).Each phrase starts with “^” so that cucumber understands the start of the step. Similarly each step ends with “$”. User can use regular expressions to pass different test data. Regular expressions take data from feature steps and passes to step definitions. The order of parameters depends how they are passed from feature file. Please refer next tutorial for project setup and mapping between feature files and java classes.

Example:
Below example is to illustrate how feature files can be implemented.

In this example we have not used any selenium API. This is to just show how cucumber works as standalone framework. Please follow next tutorial for selenium integration with cucumber.

```
public class LoginTest {
@Given("^user navigates to SOFTWARETETINGHELP.COM$")
public void navigatePage() {
system.out.println(“Cucumber executed Given statement”);
}
@When("^user logs in using Username as \"(.*)\" and Password \"(.*)\"$")
public void login(String usename,String password) {
system.out.println(“Username is:”+ usename);
 system.out.println(“Password is:”+ password);
}
@When("^click the Submit
button$")
public void clickTheSubmitButton()
{
system.out.println(“Executing When statement”)
}
@Then("^Home page should be displayed$")
public void validatePage() {
system.out.println(“Executing Then statement”)
}
@Then("^login should be successful$")
public void validateLoginSuccess() {
system.out.println(“Executing 2<sup>nd</sup> Then statement”)
}
}
```

When you execute cucumber runner class, cucumber will start reading feature file steps. For example, when you execute @smokeTest, cucumber will read **Feature** step and **Given** statement of **scenario**. As soon as cucumber finds Given statement, same **Given** statement will be searched in your java files. If same step is found in java file then cucumber executes the function specified for the same step otherwise cucumber will skip the step.

Conclusion:

In this tutorial, we have covered features of cucumber tool and its usage in real time scenario.

Cucumber is a most favorite tool for many projects as it is easy to understand, readable and contains business functionality.

In the next chapter we will cover how to setup a cucumber – java project and how to integrate Selenium WebDriver with Cucumber.

Integration of Selenium WebDriver with Cucumber – Selenium Tutorial #31

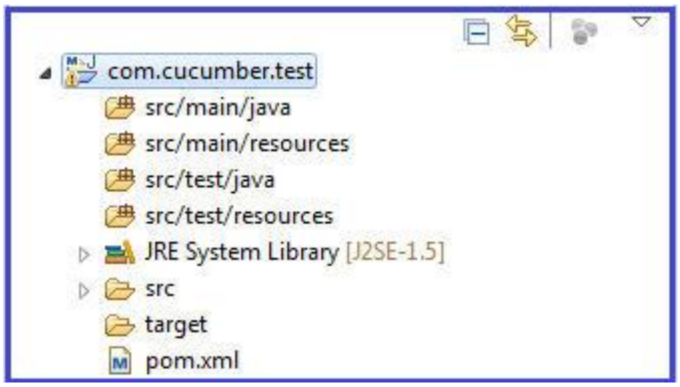
In last tutorial we discussed Cucumber tool, its usage and different features.
Moving ahead in our free Selenium online trainingseries, we will discuss how to set up a cucumber project and will discuss about integration of selenium WebDriver with Cucumber.
We will set up Cucumber project with Maven. To set up Maven in your system please refer this tutorial on Maven from the same series.

Cucumber Project Setup:

Step #1: Create New Maven Project:

Right Click -> New -> Others -> Maven -> Maven Project -> Next

Step #2: Now the project will look like this:


```
com.cucumber.test
    src/main/java
    src/main/resources
    src/test/java
    src/test/resources
    JRE System Library [J2SE-1.5]
    src
    target
    pom.xml
```

Step #3: Add below dependencies in pom.xml

```
<dependencies>
<dependency>
    <groupId>info.cukes</groupId>
    <artifactId>cucumber-java</artifactId>
    <version>1.0.2</version>
    <scope>test</scope>
  </dependency>
<dependency>
    <groupId>info.cukes</groupId>
                        <artifactId>cucumber-junit</artifactId>
    <version>1.0.2</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.10</version>
    <scope>test</scope>
</dependency>
```

```
</dependencies>
```

Step #4: Create a sample.feature file under src/test/resources.

```
@smokeTest
Feature: To test my cucumber test is running
I want to run a sample feature file.
Scenario: cucumber setup
Given sample feature file is ready
When I run the feature file
Then run should be successful
```

Step #5: Create a class under src/test/java which will implement all the steps.

```
public class stepDefinition {
    @Given("^sample feature file is ready$")
    public void givenStatment(){
        System.out.println("Given statement executed successfully");
    }
    @When("^I run the feature file$")
    public void whenStatement(){
        System.out.println("When statement execueted successfully");
    }
       @Then("^run should be successful$")
      public void thenStatment(){
          System.out.println("Then statement executed successfully");
      }
}
```

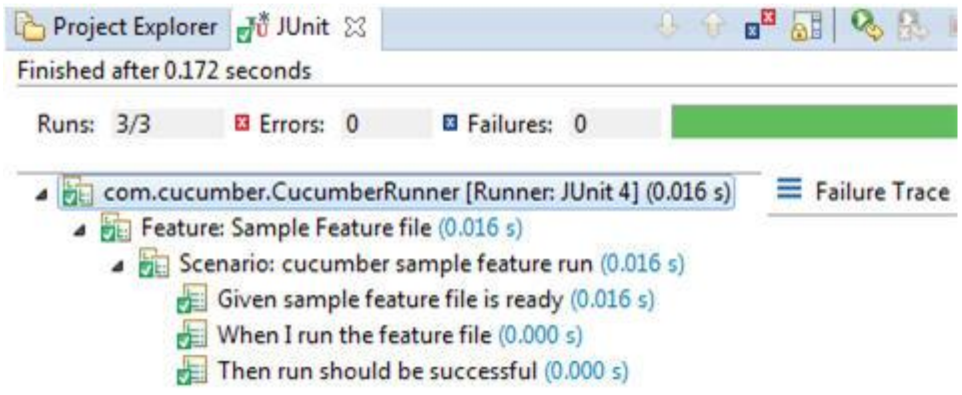
Step #6: Create a Junit runner to run the test.

```
@RunWith(Cucumber.class)
@Cucumber.Options(format={"pretty","html:reports/test-report"},tags= "@smokeTest")
public class CucumberRunner {
}
```

Provide the path of the report as given here. The reports will store in 'test-report' folder under project folder and "pretty" format specifies the type of report.

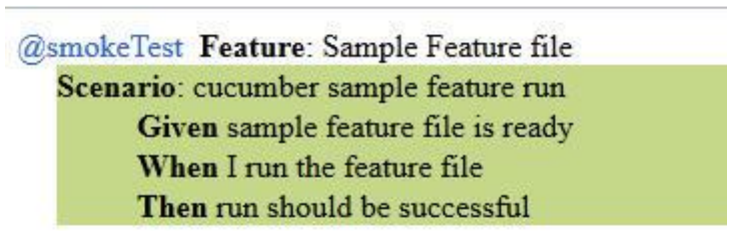
Step #7: Junit Result and Test Report:

Below is the report when the cucumber test is successful. Green bar in Junit describes the test is passed. Similarly red bar describes that test has failed.



```
Finished after 0.172 seconds
Runs: 3/3    Errors: 0    Failures: 0
com.cucumber.CucumberRunner [Runner: JUnit 4] (0.016 s)
  Feature: Sample Feature file (0.016 s)
    Scenario: cucumber sample feature run (0.016 s)
      Given sample feature file is ready (0.016 s)
      When I run the feature file (0.000 s)
      Then run should be successful (0.000 s)
```

If we want to use default reporting then navigate the path mentioned in Junit Runner. In this case we have given path as *reports->test-reports->index.html.*
Open this report in Internet Explorer or in Firefox to verify the result. Below is the sample of the report:



@smokeTest **Feature**: Sample Feature file
Scenario: cucumber sample feature run
Given sample feature file is ready
When I run the feature file
Then run should be successful

Cucumber and Selenium WebDriver:
Cucumber framework can be used to test the web based applications along with Selenium WebDriver. The test cases are written in simple feature files which are easily understood by managers, non-technical stake holders and business analysts. And those feature file steps are implemented in step definition file. If you are using maven then you have to add dependencies for Cucumber and WebDriver.

So here is the sample test case we have implemented using Cucumber and WebDriver. As given below, the scenario in feature file is self-explanatory.

Feature: Login Feature File
@selenium
Scenario: Login scenario test for Gmail
Given navigate to gmail page
When user logged in using username as "userA" and password as "password"
Then home page should be displayed
<u>WebDriver Implementation in Cucumber stepDefinitions:</u>

```
public class stepDefinition {
WebDriver dr;
@Given("^navigate to gmail page$")
public void navigate(){
    dr=new FirefoxDriver();
    dr.get("http://www.gmail.com");
    }
@When ("^user logged in using username as \"(.*)\" and password as \"(.*)\"$")
public void login(String username,String password){
                        dr.findElement(By.xpath("//*[@id='Email']")).sendKeys(username);
     dr.findElement(By.xpath("//*[@id='Passwd']")).sendKeys(password);
     dr.findElement(By.xpath("//*[@id='signIn']")).click();
     dr.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);
     }
@Then("^home page should be displayed$")
public void verifySuccessful(){
    String expectedText="Gmail";
    String actualText=      dr.findElement(By.xpath("//*[@id='gbq1']/div/a/span")).getText();
    Assert.assertTrue("Login not successful",expectedText.equals(actualText));
    }
}
```

In this test we have used the Firefox as the browser to test the Gmail login functionality.
Clearly WebDriver object is a class variable and used across the class.
Given statement initializes the browser and navigates to the page.
When statement logs into the application using username as "userA" and password as "password". Both the values 'username' and 'password' are passed from feature file and both the values to be used in the same order.
Then Statement only validates the conditions after logging into the application.
This is a sample test describing the usage of Cucumber and Selenium. You can create multilayer architecture depending upon your project requirement.

Conclusion:
In this Tutorial we have covered most of the Cucumber concepts which includes Cucumber features and its usage along with WebDriver.

This reduces the complexity of code which is written to design the traditional frameworks like Keyword Driven and Hybrid Framework. Cucumber is used in most of the project where people follow agile methodology as Behavior Driven Development is an Agile Software practice.

Next Tutorial #32: We have now completed all technical tutorials from this Selenium training series. Next, we will post about few important general topics like **'effort estimation for Selenium projects' and 'Selenium interview questions with answers'.**
Please post your queries regarding to Cucumber.

Selenium Tips and Interview Preparation

7 Factors Affecting Test Estimation of Selenium Automation Project – Selenium Tutorial #32

In last couple of Selenium tutorials we learned about automation testing using Cucumber and Selenium tool. We also discussed about integration of selenium WebDriver with Cucumber.
In this tutorial we will discuss about different **factors affecting effort estimation of Selenium automation**. Planning and estimation are two most important aspect of a software development life cycle.
I personally feel that in software industry, there are **no bullet proof methods** of doing anything. Since every project is exclusive and have different sets of complexity and environmental factors, implementing the estimation and planning strategy should be a collaborative effort of the individual teams with proper interventions of seniors and management support.

Before you begin with estimating any project, it is imperial to understand each and every phase that your project will be going through, so that you can give a correct and a justified estimation.
Estimation can not only be done for the manual testing process, but in this era of automation, estimation techniques are applied to test automation as well. Now Selenium gaining a momentum and popularity in the market, I am trying to write about some factors which should be taken into consideration while estimating a Selenium project.

Let's Start!!

I am assuming that we are starting the Automation initiative from the scratch and that we have no ready-made framework available.

Factors affecting estimation of selenium automation
The various factors which effect and which you should consider for estimation of "Selenium" specific project are explained below:

#1 Scope of the project
Scope typically means identifying the correct test cases for automation. Apply "Divide & rule" strategy to accomplish it. Break your application in small chunks or modules and analyze each of them to come up with the appropriate test cases for automation.

The steps involved are:

1. Identify the various factors which will form the basis of identifying the candidate test cases.

2. Break the application into smaller modules
3. Analyze each module to identify the candidate test cases
4. Calculate ROI

For more details of how to identify the correct test case, please see my previous paper: Selection of correct test cases for Automation

#2 Complexity of the application

Steps involved here are:

1. Determine the Size the application based on the number of test cases that needs to be automated.
2. Size complexity through Fibonacci series.
3. Identify the verification point and check point of each test case

Here we have to establish the definition of big / medium and small sized application. This definition differs from an individual / group perspective. How you classify your application, depends can also be dependent upon the number of test cases.

For example:

If your application has 300 – 500 test cases to automate, you can consider it as small sized application. If the test cases are over 1500, it can be classified as complex. This factor can be different for different application. For some, 1500 test cases to automate can be considered as small / medium scaled. So once you have identified the exact number of test cases, scale it to small / medium or large. Your strategy towards estimating the effort will hugely dependent on these criteria.

You have to also consider the different check points and verification points for your test case. A test case can have more than 1 check point but will have only 1 verification point. In case you have more than 1 verification point, it is recommended to bifurcate into separate test cases. This will also ease your maintenance and enhancement of your test suite.

#3 Use of supporting tools / technologies

Steps involved here are:

1. Identify the framework and automation needs
2. Based on the needs, analyze and identify the tools to be used.
3. Identify the dependencies / implications of using the tool.

Selenium alone is not sufficient to build a framework or complete the automation. Selenium (Web driver) will only script the test case, but there are other tasks as well, like reporting the result, tracking the logs, taking screen shots etc.

To achieve these you need separate tools that will be integrated with your framework. So it is important here to identify these supporting entities which will best suite your requirement and will help to get a positive ROI

#4 Implementing the Framework

Here comes the tricky part J the steps involved are!!

1. Identify the input (pattern in which data is fed in to script) and output (reports / test results) of your automation suite.
2. Design your input files. This may range from a simple text file to complex excel file. It is basically the file which will have your test data.
3. Design the folder structure based on your input parameters and
4. Implement the reporting feature (either in some excel file or using any tool like ReportNG)
5. Determine / implement logger in your frame work
6. Implement the build tool in your framework
7. Implement the unit test framework (Junit or TestNG)

There are many other requirements apart from just scripting in test automation with Selenium, like reading the data from a file, reporting / tracking the test results, tracking logs , trigger the scripts based on the input conditions and environment etc. So we need a structure that will take care of all these scripts. This structure is nothing but your Framework.

Web applications are complex by nature because it involves lots of supporting tools and technology to implement. In a similar way, implementing framework in Selenium is also tricky (I will not say complex) as it involves other tools to integrate. Since we know Selenium is NOT a tool but actually a collection / group of jar files, it is configured and not “Installed”, Selenium itself is not strong enough to build a complex framework. It requires a list of third party tools for building a framework.

We have to remember here that there is nothing “Ready-made” in Selenium. For everything, we have to code, so provisions in estimation should be there for googling the errors and troubleshooting.

Here we have to understand that that Framework building is the most important aspect of your Automation effort. If your framework is rock solid, maintenance and enhancement becomes easier specially in the era of Agile, if your framework is good, you can integrate your tests in all the sprints easily.

I won’t be wrong if I say that this particular factor of designing the Framework should be the most important aspect of estimation. If needed (like in complex application) this factor should be again broken down into a separate WBS and estimation should be done.

#5 Learning & Training

Learning Selenium is a bit different than learning any other automation tool. It basically involves learning a programming language than just a scripting language (though script language helps while building a framework , like you want to write a script that would invoke your automated scripts after doing the environment setting changes).

In case we are combining WebDriver with java, I would say that if one is well versed with core java, they are in a very good shape to start with selenium automation.

Along with learning java, provisions should be there to learn other technologies like ANT / Maven(for building), TestNG/jUnit (unit test framework), Log4J(for Logging), reporting (for reporting) etc. this list may grow based on the level of framework. The more this list grows, the more time it would take.

If the management has decided to go with selenium, these learning activities can be done parallel with the planning activity. Because there is no limit to learn these technologies, it is suggested to have a definite plan (syllabus) ready for the team so that they can initiated their learning process in a definite direction and everybody is in the same page.

Practically speaking, we testers do not have a very much keen in learning a full-fledged programming language and we feel this is developers piece of cake. But now we have to change this mentality and should consider learning the programming language to be equally important as learning new testing process. This will not only increase tester’s knowledge about the language and automation but also will give a chance to understand how the application works internally which may increase their scope to find new bugs.

#6 Environment setup

Environment set up deals with (not limited to):-

- Setting up the code in the test environment
- Setting up code in production environment
- Writing scripts for triggering the automated tests.
- Developing the logic for reporting
- Establishing the frequency of running the scripts and developing logic for its implementation
- Creating text / excel files for entering the test data and test cases
- Creating property files for tracking the environments and credentials

#7 Coding / scripting and review

Before you actually start writing your tests, there are 2 prerequisites:

1. Candidate test cases should be handy
2. Framework is ready

Identify different actions that your web application does. It can be simple actions like navigation, clicking, entering text; or a complex action like connect to database, handle flash or Ajax. Take one test case at a time, and identify what all action that particular test case does and estimate hours accordingly for per test case. The sum of all the hours for the entire test suite will give you the exact number.

Provision should be there for Review as well. The reviews are simple the code review which can be done either by peer or a developer. Pair programming is the best option which is quick, but if it is not possible, based on the available resources or organizations review strategy, hours should be allocated for it.

More details about each factor affecting estimation:

Factor #1: Scope

Meaning: Identifying the candidate test cases for automation through the ROI

Steps Involved:

1. Identify the various factors which will form the basis of identifying the candidate test cases.
2. Break the application into smaller modules
3. Analyze each module to identify the candidate test cases
4. Calculate the ROI

Deliverable: List of test cases that needs to automated.

Remarks: It is important to freeze your scope once you go ahead with other steps of estimation.

Factor #2: Complexity

Meaning: Establish the definition of big / medium and small sized application.

Steps Involved:

1. Size the application based on the number of test cases that needs to be automated.
2. Size complexity through Fibonacci series.
3. Identify the verification point and check point of each test case.

Deliverable: Size of the application – Small, medium or Big.

Number of test cases and their corresponding checkpoint and verification point.

Remarks: Recommended – A test case can have multiple check point but only 1 verification point. If a test case has more than 1 verification point, it should be bifurcated into a separate test case.

Factor #3: Supporting tools

Meaning: Selenium itself is not strong enough to build a complex framework. It requires a list of third party tools for building a framework.

Steps Involved:

1. Finalized IDE

2. Finalized unit test tool
3. Finalized logger
4. Finalized reporting tool
5. Finalized build tool

Deliverable: List of tools needed to create the framework.

Remarks:

Examples:

- Eclipse / RAD – as IDE
- Ant / Maven – As build tool
- jUnit / TestNG – as unit test framework
- Log4j – as Logger
- ReportiNG – as reporting tool
- Text files – for tracking the environments / credentials
- Excel files – for tracking the test data
- Perl / Python – for setting up environment and triggering the test scripts.

Factor #4: Implementing Framework

Meaning: Creation of structure

Steps Involved:

1. Design your input files.
2. Design the folder structure
3. Determine / implement logger in your frame work
4. Implement the build tool in your framework
5. Implement the unit test framework

Deliverable:

- Framework and folder structure created in the IDE.
- Excel sheets containing your input data
- Property files containing environment related data and credentials.

Remarks: This is the most crucial step. It is advisable to include some buffer time while estimating because some time trouble shooting take more time than expected.

Factor #5: Environment set up

Meaning: Deals with code set up and downloading / preparing for the code deployment

Steps Involved:

1. Prepare the input file and reporting
2. Create the triggering script

Deliverable: Environment ready

Remarks: We should try to build our framework in such a way that with least hassle, our code is deployed in to the said environment / box.
I should not be wrong if I say that with minimal entries into our text files (which have the url and credentials) our code should be ready to run and ROCK!

Factor #6: Learning & training

Meaning: Learning a programming language and other supporting technologies

Steps Involved: Prepare a plan as per your automation needs and share it with the team and encourage them to learn and proceed as per the syllabus.

Deliverable: Training Plan and its tracker which will track the progress of the team.

Remarks: Emphasis should be on building logics rather learning syntax.

Factor #7: Coding / scripting and Review

Meaning: Writing the actual test scripts and reviewing them

Steps Involved:

1. Test cases and framework is ready.
2. Take / divide the test cases and convert it into automated scripts and track your progress

Deliverable: Automated test scripts

Remarks: Whole team should participate in writing the test scripts using the implemented framework. So while estimating, efforts from the whole team should be taken into consideration.

Conclusion:

Having said about all these points, do not forget to include Management overhead and some buffer time in your final Selenium automation estimation. The best and the proven way to do any estimation is to follow the WBS (Work Break down Structure) mechanism. This is straight forward and serves the purpose of implementing the automation estimation needs.

The factors mentioned above are the ones based on my experience, but there can be other entities as well which might affect the strategy.

The thumb rule here is **"Identify certain criteria, divide your modules or test case on those criteria; and scale it".** Based on your scaled figure – you can come to an accurate estimation.

Next Tutorial #33: We will be concluding our most comprehensive Selenium online training free tutorials series with last tutorial i.e. "**Selenium testing interview questions with answers**".

Let us know if you have any other tips for effort estimation of Selenium projects.

50 Most Popularly Asked Selenium Interview Questions and Answers – Selenium Tutorial #33

In this tutorial, we have listed the **50 most popularly asked Selenium interview questions including Selenium WebDriver interview questions.**
A quick note about this Selenium article series before we move to this last tutorial:
This is the last tutorial in our Selenium online training series of 30+ comprehensive tutorials. I hope you all enjoyed these tutorials and started learning from it. If you are new here please head over to this very first tutorial in this training series.

Top 50 Selenium Interview Questions and Answers:

Q #1) What is Automation Testing?
Automation testing or Test Automation is a process of automating the manual process to test the application/system under test. Automation testing involves use to a separate testing tool which lets you create test scripts which can be executed repeatedly and doesn't require any manual intervention.

Q #2) What are the benefits of Automation Testing?
Benefits of Automation testing are:

1. Supports execution of repeated test cases
2. Aids in testing a large test matrix
3. Enables parallel execution
4. Encourages unattended execution
5. Improves accuracy thereby reducing human generated errors
6. Saves time and money

Q #3) Why should Selenium be selected as a test tool?
Selenium

1. is free and open source
2. have a large user base and helping communities
3. have cross Browser compatibility (Firefox, chrome, Internet Explorer, Safari etc.)
4. have great platform compatibility (Windows, Mac OS, Linux etc.)
5. supports multiple programming languages (Java, C#, Ruby, Python, Pearl etc.)
6. has fresh and regular repository developments

7. supports distributed testing

Q #4) What is Selenium? What are the different Selenium components?

Selenium is one of the most popular automated testing suites. Selenium is designed in a way to support and encourage automation testing of functional aspects of web based applications and a wide range of browsers and platforms. Due to its existence in the open source community, it has become one of the most accepted tools amongst the testing professionals.

Selenium is not just a single tool or a utility, rather a package of several testing tools and for the same reason it is referred to as a Suite. Each of these tools is designed to cater different testing and test environment requirements.

The suite package constitutes of the following sets of tools:

- **Selenium Integrated Development Environment (IDE)** – Selenium IDE is a record and playback tool. It is distributed as a Firefox Plugin.
- **Selenium Remote Control (RC)** – Selenium RC is a server that allows user to create test scripts in a desired programming language. It also allows executing test scripts within the large spectrum of browsers.
- **Selenium WebDriver** – WebDriver is a different tool altogether that has various advantages over Selenium RC. WebDriver directly communicates with the web browser and uses its native compatibility to automate.
- **Selenium Grid** – Selenium Grid is used to distribute your test execution on multiple platforms and environments concurrently.

Q #5) What are the testing types that can be supported by Selenium?

Selenium supports the following types of testing:

1. Functional Testing
2. Regression Testing

Q #6) What are the limitations of Selenium?

Following are the limitations of Selenium:

- Selenium supports testing of only web based applications
- Mobile applications cannot be tested using Selenium
- Captcha and Bar code readers cannot be tested using Selenium
- Reports can only be generated using third party tools like TestNG or Junit.
- As Selenium is a free tool, thus there is no ready vendor support though the user can find numerous helping communities.

- User is expected to possess prior programming language knowledge.

Q #7) What is the difference between Selenium IDE, Selenium RC and WebDriver?

Feature	Selenium IDE	Selenium RC	WebDriver
Browser Compatibility	Selenium IDE comes as a Firefox plugin, thus it supports only Firefox	Selenium RC supports a varied range of versions of Mozilla Firefox, Google Chrome, Internet Explorer and Opera	WebDriver supports a varied range of versions of Mozilla Firefox, Google Chrome, Internet Explorer and Opera. Also supports HtmlUnitDriver which is a GUI less or headless browser.
Record and Playback	Selenium IDE supports record and playback feature	Selenium RC doesn't supports record and playback feature	WebDriver doesn't support record and playback feature
Server Requirement	Selenium IDE doesn't require any server to be started before executing the test scripts	Selenium RC requires server to be started before executing the test scripts	WebDriver doesn't require any server to be started before executing the test scripts
Architecture	Selenium IDE is a Javascript based framework	Selenium RC is a JavaScript based Framework	WebDriver uses the browser's native compatibility to automation
Object Oriented	Selenium IDE is not an object oriented tool	Selenium RC is semi object oriented tool	WebDriver is a purely object oriented tool
Dynamic Finders (for locating web elements on a webpage)	Selenium IDE doesn't support dynamic finders	Selenium RC doesn't support dynamic finders	WebDriver supports dynamic finders

Feature	Selenium IDE	Selenium RC	WebDriver
Handling Alerts, Navigations, Dropdowns	Selenium IDE doesn't explicitly provides aids to handle alerts, navigations, dropdowns	Selenium RC doesn't explicitly provides aids to handle alerts, navigations, dropdowns	WebDriver offers a wide range of utilities and classes that helps in handling alerts, navigations, and dropdowns efficiently and effectively.
WAP (iPhone/Android) Testing	Selenium IDE doesn't support testing of iPhone/Andriod applications	Selenium RC doesn't support testing of iPhone/Andriod applications	WebDriver is designed in a way to efficiently support testing of iPhone/Android applications. The tool comes with a large range of drivers for WAP based testing. For example, AndroidDriver, iPhoneDriver
Listener Support	Selenium IDE doesn't support listeners	Selenium RC doesn't support listeners	WebDriver supports the implementation of Listeners
Speed	Selenium IDE is fast as it is plugged in with the web-browser that launches the test. Thus, the IDE and browser communicates directly	Selenium RC is slower than WebDriver as it doesn't communicates directly with the browser; rather it sends selenese commands over to Selenium Core which in turn	WebDriver communicates directly with the web browsers. Thus making it much faster.

Feature	Selenium IDE	Selenium RC	WebDriver
		communicates with the browser.	

Q #8) When should I use Selenium IDE?

Selenium IDE is the simplest and easiest of all the tools within the Selenium Package. Its record and playback feature makes it exceptionally easy to learn with minimal acquaintances to any programming language. Selenium IDE is an ideal tool for a naïve user.

Q #9) What is Selenese?

Selenese is the language which is used to write test scripts in Selenium IDE.

Q #10) What are the different types of locators in Selenium?

Locator can be termed as an address that identifies a web element uniquely within the webpage. Thus, to identify web elements accurately and precisely we have different types of locators in Selenium:

- ID
- ClassName
- Name
- TagName
- LinkText
- PartialLinkText
- Xpath
- CSS Selector
- DOM

Q #11) What is difference between assert and verify commands?

Assert: Assert command checks whether the given condition is true or false. Let's say we assert whether the given element is present on the web page or not. If the condition is true then the program control will execute the next test step but if the condition is false, the execution would stop and no further test would be executed.
Verify: Verify command also checks whether the given condition is true or false. Irrespective of the condition being true or false, the program execution doesn't halts i.e. any failure during verification would not stop the execution and all the test steps would be executed.

Q #12) What is an Xpath?

Xpath is used to locate a web element based on its XML path. XML stands for Extensible Markup Language and is used to store, organize and transport arbitrary data. It stores data in a key-value pair which is very much similar to HTML tags. Both being markup languages and since they fall under the same umbrella, Xpath can be used to locate HTML elements.

The fundamental behind locating elements using Xpath is the traversing between various elements across the entire page and thus enabling a user to find an element with the reference of another element.

Q #13) What is the difference between "/" and "//" in Xpath?

Single Slash "/" – Single slash is used to create Xpath with absolute path i.e. the xpath would be created to start selection from the document node/start node.

Double Slash "//" – Double slash is used to create Xpath with relative path i.e. the xpath would be created to start selection from anywhere within the document.

Q #14) What is Same origin policy and how it can be handled?

The problem of same origin policy disallows to access the DOM of a document from an origin that is different from the origin we are trying to access the document.

Origin is a sequential combination of scheme, host and port of the URL. For example, for a URL http://http://www.softwaretestinghelp.com/resources/, the origin is a combination of http, softwaretestinghelp.com, 80 correspondingly.

Thus the Selenium Core (JavaScript Program) cannot access the elements from an origin that is different from where it was launched. For Example, if I have launched the JavaScript Program from "http://www.softwaretestinghelp.com", then I would be able to access the pages within the same domain such as "http://www.softwaretestinghelp.com/resources" or "http://www.softwaretestinghelp.com/istqb-free-updates/". The other domains like google.com, seleniumhq.org would no more be accessible.

So, In order to handle same origin policy, Selenium Remote Control was introduced.

Q #15) When should I use Selenium Grid?

Selenium Grid can be used to execute same or different test scripts on multiple platforms and browsers concurrently so as to achieve distributed test execution, testing under different environments and saving execution time remarkably.

Q #16) What do we mean by Selenium 1 and Selenium 2?

Selenium RC and WebDriver, in a combination are popularly known as Selenium 2. Selenium RC alone is also referred as Selenium 1.

Q #17) Which is the latest Selenium tool?

WebDriver

Q #18) How do I launch the browser using WebDriver?

The following syntax can be used to launch Browser:

*WebDriver driver = **new** FirefoxDriver();*

*WebDriver driver = **new** ChromeDriver();*

*WebDriver driver = **new** InternetExplorerDriver();*

Q #19) What are the different types of Drivers available in WebDriver?

The different drivers available in WebDriver are:

- FirefoxDriver
- InternetExplorerDriver
- ChromeDriver
- SafariDriver
- OperaDriver
- AndroidDriver
- IPhoneDriver
- HtmlUnitDriver

Q #20) What are the different types of waits available in WebDriver?

There are two types of waits available in WebDriver:

1. Implicit Wait
2. Explicit Wait

Implicit Wait: Implicit waits are used to provide a default waiting time (say 30 seconds) between each consecutive test step/command across the entire test script. Thus, subsequent test step would only execute when the 30 seconds have elapsed after executing the previous test step/command.

Explicit Wait: Explicit waits are used to halt the execution till the time a particular condition is met or the maximum time has elapsed. Unlike Implicit waits, explicit waits are applied for a particular instance only.

Q #21) How to type in a textbox using Selenium?

User can use sendKeys("String to be entered") to enter the string in the textbox.

Syntax:

WebElement username = drv.findElement(By.id("Email"));

// entering username

username.sendKeys("sth");

Q #22) How can you find if an element in displayed on the screen?

WebDriver facilitates the user with the following methods to check the visibility of the web elements. These web elements can be buttons, drop boxes, checkboxes, radio buttons, labels etc.

1. isDisplayed()
2. isSelected()
3. isEnabled()

Syntax:

isDisplayed():

boolean *buttonPresence = driver.findElement(By.id("gbqfba")).isDisplayed();*

isSelected():

boolean *buttonSelected = driver.findElement(By.id("gbqfba")).isDisplayed();*

isEnabled():

boolean *searchIconEnabled = driver.findElement(By.id("gbqfb")).isEnabled();*

Q #23) How can we get a text of a web element?

Get command is used to retrieve the inner text of the specified web element. The command doesn't require any parameter but returns a string value. It is also one of the extensively used commands for verification of messages, labels, errors etc displayed on the web pages.

Syntax:

String Text = driver.findElement(By.id("Text")).getText();

Q #24) How to select value in a dropdown?

Value in the drop down can be selected using WebDriver's Select class.

Syntax:

selectByValue:

Select selectByValue = ***new*** *Select(driver.findElement(By.id("SelectID_One")));*

selectByValue.selectByValue("greenvalue");

selectByVisibleText:

Select selectByVisibleText = ***new*** *Select (driver.findElement(By.id("SelectID_Two")));*

selectByVisibleText.selectByVisibleText("Lime");

selectByIndex:

Select selectByIndex = ***new*** *Select(driver.findElement(By.id("SelectID_Three")));*

selectByIndex.selectByIndex(2);

Q #25) What are the different types of navigation commands?

Following are the navigation commands:

navigate().back() – The above command requires no parameters and takes back the user to the previous webpage in the web browser's history.

Sample code:

driver.navigate().back();

navigate().forward() – This command lets the user to navigate to the next web page with reference to the browser's history.

Sample code:

driver.navigate().forward();

navigate().refresh() – This command lets the user to refresh the current web page there by reloading all the web elements.

Sample code:

driver.navigate().refresh();

navigate().to() – This command lets the user to launch a new web browser window and navigate to the specified URL.

Sample code:

driver.navigate().to("https://google.com");

Q #26) How to click on a hyper link using linkText?

driver.findElement(By.linkText("Google")).click();

The command finds the element using link text and then click on that element and thus the user would be re-directed to the corresponding page.

The above mentioned link can also be accessed by using the following command.

----

driver.findElement(By.partialLinkText("Goo")).click();

The above command find the element based on the substring of the link provided in the parenthesis and thus partialLinkText() finds the web element with the specified substring and then clicks on it.

Q #27) How to handle frame in WebDriver?

An inline frame acronym as iframe is used to insert another document with in the current HTML document or simply a web page into a web page by enabling nesting.

Select iframe by id

driver.switchTo().frame("ID of the frame");

Locating iframe using tagName

driver.switchTo().frame(driver.findElements(By.tagName("iframe").get(0));

Locating iframe using index

frame(index)

driver.switchTo().frame(0);

frame(Name of Frame)

driver.switchTo().frame("name of the frame");

frame(WebElement element)

Select Parent Window

driver.switchTo().defaultContent();

Q #28) When do we use findElement() and findElements()?

findElement(): findElement() is used to find the first element in the current web page matching to the specified locator value. Take a note that only first matching element would be fetched.

Syntax:

WebElement element = driver.findElements(By.xpath("//div[@id='example']//ul//li"));

findElements(): findElements() is used to find all the elements in the current web page matching to the specified locator value. Take a note that all the matching elements would be fetched and stored in the list of WebElements.

Syntax:

List <WebElement> elementList = driver.findElements(By.xpath("//div[@id='example']//ul//li"));

Q #29) How to find more than one web element in the list?

At times, we may come across elements of same type like multiple hyperlinks, images etc arranged in an ordered or unordered list. Thus, it makes absolute sense to deal with such elements by a single piece of code and this can be done using WebElement List.

Sample Code

```
// Storing the list
List <WebElement> elementList = driver.findElements(By.xpath("//div[@id='example']//ul//li"));
// Fetching the size of the list
int listSize = elementList.size();
for (int i=0; i<listSize; i++)
{
// Clicking on each service provider link
serviceProviderLinks.get(i).click();
 // Navigating back to the previous page that stores link to service providers
driver.navigate().back();
}
```

Q #30) What is the difference between driver.close() and driver.quit command?

close(): WebDriver's close() method closes the web browser window that the user is currently working on or we can also say the window that is being currently accessed by the WebDriver. The command neither requires any parameter nor does is return any value.

quit(): Unlike close() method, quit() method closes down all the windows that the program has opened. Same as close() method, the command neither requires any parameter nor does is return any value.

Q #31) Can Selenium handle windows based pop up?

Selenium is an automation testing tool which supports only web application testing. Therefore, windows pop up cannot be handled using Selenium.

Q #32) How can we handle web based pop up?

WebDriver offers the users with a very efficient way to handle these pop ups using Alert interface. There are the four methods that we would be using along with the Alert interface.

- void dismiss() – The accept() method clicks on the “Cancel” button as soon as the pop up window appears.
- void accept() – The accept() method clicks on the “Ok” button as soon as the pop up window appears.
- String getText() – The getText() method returns the text displayed on the alert box.
- void sendKeys(String stringToSend) – The sendKeys() method enters the specified string pattern into the alert box.

Syntax:

```
// accepting javascript alert
        Alert alert = driver.switchTo().alert();
alert.accept();
```

Q #33) How can we handle windows based pop up?

Selenium is an automation testing tool which supports only web application testing, that means, it doesn’t support testing of windows based applications. However Selenium alone can’t help the situation but along with some third party intervention, this problem can be overcome. There are several third party tools available for handling window based pop ups along with the selenium like AutoIT, Robot class etc.

Q #34) How to assert title of the web page?

```
//verify the title of the web page
assertTrue(“The title of the window is incorrect.”,driver.getTitle().equals(“Title of the page”));
```

Q #35) How to mouse hover on a web element using WebDriver?

WebDriver offers a wide range of interaction utilities that the user can exploit to automate mouse and keyboard events. Action Interface is one such utility which simulates the single user interactions.

Thus, In the following scenario, we have used Action Interface to mouse hover on a drop down which then opens a list of options.

Sample Code:

```
// Instantiating Action Interface
Actions actions=new Actions(driver);
// howering on the dropdown
actions.moveToElement(driver.findElement(By.id("id of the dropdown"))).perform();
// Clicking on one of the items in the list options
WebElement subLinkOption=driver.findElement(By.id("id of the sub link"));
subLinkOption.click();
```

Q #36) How to retrieve css properties of an element?

The values of the css properties can be retrieved using a get() method:

Syntax:

driver.findElement(By.id("id")).getCssValue("name of css attribute");

driver.findElement(By.id("id")).getCssValue("font-size");

Q #37) How to capture screenshot in WebDriver?

```
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import java.io.File;
import java.io.IOException;
import org.apache.commons.io.FileUtils;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.openqa.selenium.WebDriver;
                    import org.openqa.selenium.firefox.FirefoxDriver;

public class CaptureScreenshot {
    WebDriver driver;
    @Before
    public void setUp() throws Exception {
        driver = new FirefoxDriver();
        driver.get("https://google.com");
   }
   @After
   public void tearDown() throws Exception {
        driver.quit();
   }

   @Test
   public void test() throws IOException
{
        // Code to capture the screenshot
File scrFile = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);
        // Code to copy the screenshot in the desired location
FileUtils.copyFile(scrFile, newFile("C:\\CaptureScreenshot\\google.jpg"));
    }
}
```

Q #38) What is Junit?

Junit is a unit testing framework introduced by Apache. Junit is based on Java.

Q #39) What are Junit annotations?

Following are the Junit Annotations:

- **@Test:** Annotation lets the system know that the method annotated as @Test is a test method. There can be multiple test methods in a single test script.

- **@Before:** Method annotated as @Before lets the system know that this method shall be executed every time before each of the test method.
- **@After:** Method annotated as @After lets the system know that this method shall be executed every time after each of the test method.
- **@BeforeClass:** Method annotated as @BeforeClass lets the system know that this method shall be executed once before any of the test method.
- **@AfterClass:** Method annotated as @AfterClass lets the system know that this method shall be executed once after any of the test method.
- **@Ignore:** Method annotated as @Ignore lets the system know that this method shall not be executed.

Q #40) What is TestNG and how is it better than Junit?

TestNG is an advance framework designed in a way to leverage the benefits by both the developers and testers. With the commencement of the frameworks, JUnit gained an enormous popularity across the Java applications, Java developers and Java testers with remarkably increasing the code quality. Despite being easy to use and straightforward, JUnit has its own limitations which give rise to the need of bringing TestNG into the picture. TestNG is an open source framework which is distributed under the Apache software License and is readily available for download.

TestNG with WebDriver provides an efficient and effective test result format that can in turn be shared with the stake holders to have a glimpse on the product's/application's health thereby eliminating the drawback of WebDriver's incapability to generate test reports. TestNG has an inbuilt exception handling mechanism which lets the program to run without terminating unexpectedly.

There are various advantages that make TestNG superior to JUnit. Some of them are:

- Added advance and easy annotations
- Execution patterns can set
- Concurrent execution of test scripts
- Test case dependencies can be set

Q #41) How to set test case priority in TestNG?

Setting Priority in TestNG

Code Snippet

```
package TestNG;
import org.testng.annotations.*;
public class SettingPriority {
    @Test(priority=0)
    public void method1() {
    }
    @Test(priority=1)
    public void method2() {
    }
        @Test(priority=2)
    public void method3() {
    }
}
```

Test Execution Sequence:

1. Method1
2. Method2
3. Method3

Q #42) What is a framework?

Framework is a constructive blend of various guidelines, coding standards, concepts, processes, practices, project hierarchies, modularity, reporting mechanism, test data injections etc. to pillar automation testing.

Q #43) What are the advantages of Automation framework?

Advantage of Test Automation framework

- Reusability of code
- Maximum coverage
- Recovery scenario
- Low cost maintenance
- Minimal manual intervention
- Easy Reporting

Q #44) What are the different types of frameworks?

Below are the different types of frameworks:

1. **Module Based Testing Framework:** The framework divides the entire "Application Under Test" into number of logical and isolated modules. For each module, we create a separate and independent test script. Thus, when these test scripts taken together builds a larger test script representing more than one module.
2. **Library Architecture Testing Framework:** The basic fundamental behind the framework is to determine the common steps and group them into functions under a library and call those functions in the test scripts whenever required.
3. Data Driven Testing Framework: Data Driven Testing Framework helps the user segregate the test script logic and the test data from each other. It lets the user store the test data into an external database. The data is conventionally stored in "Key-Value" pairs. Thus, the key can be used to access and populate the data within the test scripts.
4. **Keyword Driven Testing Framework:** The Keyword driven testing framework is an extension to Data driven Testing Framework in a sense that it not only segregates the test data from the scripts, it also keeps the certain set of code belonging to the test script into an external data file.

5. **Hybrid Testing Framework:** Hybrid Testing Framework is a combination of more than one above mentioned frameworks. The best thing about such a setup is that it leverages the benefits of all kinds of associated frameworks.
6. **Behavior Driven Development Framework:** Behavior Driven Development framework allows automation of functional validations in easily readable and understandable format to Business Analysts, Developers, Testers, etc.

Q #45) How can I read test data from excels?

Test data can efficiently be read from excel using JXL or POI API. See detailed tutorial here.

Q #46) What is the difference between POI and jxl jar?

#	JXL jar	POI jar
1	JXL supports ".xls" format i.e. binary based format. JXL doesn't support Excel 2007 and ".xlsx" format i.e. XML based format	POI jar supports all of these formats
2	JXL API was last updated in the year 2009	POI is regularly updated and released
3	The JXL documentation is not as comprehensive as that of POI	POI has a well prepared and highly comprehensive documentation
4	JXL API doesn't support rich text formatting	POI API supports rich text formatting
5	JXL API is faster than POI API	POI API is slower than JXL API

Q #47) What is the difference between Selenium and QTP?

Feature	Selenium	Quick Test Professional (QTP)
Browser Compatibility	Selenium supports almost all the popular browsers like Firefox, Chrome, Safari, Internet Explorer, Opera etc	QTP supports Internet Explorer, Firefox and Chrome. QTP only supports Windows Operating System
Distribution	Selenium is distributed as an open source tool and is freely available	QTP is distributed as a licensed tool and is commercialized
Application under Test	Selenium supports testing of only web based	QTP supports testing of both the web based

Feature	Selenium	Quick Test Professional (QTP)
	applications	application and windows based application
Object Repository	Object Repository needs to be created as a separate entity	QTP automatically creates and maintains Object Repository
Language Support	Selenium supports multiple programming languages like Java, C#, Ruby, Python, Perl etc	QTP supports only VB Script
Vendor Support	As Selenium is a free tool, user would not get the vendor's support in troubleshooting issues	Users can easily get the vendor's support in case of any issue

Q #48) Can WebDriver test Mobile applications?

WebDriver cannot test Mobile applications. WebDriver is a web based testing tool, therefore applications on the mobile browsers can be tested.

Q #49) Can captcha be automated?

No, captcha and bar code reader cannot be automated.

Q #50) What is Object Repository? How can we create Object Repository in Selenium?

Object Repository is a term used to refer to the collection of web elements belonging to Application Under Test (AUT) along with their locator values. Thus, whenever the element is required within the script, the locator value can be populated from the Object Repository. Object Repository is used to store locators in a centralized location instead of hard coding them within the scripts.

In Selenium, objects can be stored in an excel sheet which can be populated inside the script whenever required.

That's all for now.

Hope in this article you will find answers to most frequently asked Selenium and WebDriver Interview questions. The answers provided here are also helpful for understanding the Selenium basics and advanced WebDriver topics.

Do you have any Selenium Interview questions that are not answered here? *Please let us know in comments below and we will try to answer all.*
=> This finishes not just this article but our complete Selenium training series. Check list of ALL 30+ tutorials listed on this page. Please let us know your comments and questions.

****

Source:
http://www.softwaretestinghelp.com/category/selenium-tutorials/

As of 2017 Feb 09